Departamento de Ciencias da Computación e
Tecnoloxías da Información
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRDAO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE INFORMACIÓN

Aplicación web de gestión de información geográfica para personas con movilidad reducida

Estudiante: Juan Manuel Fontenla Valiña
Dirección: Miguel Ángel Rodríguez Luaces
Alejandro Cortiñas Álvarez

A Coruña, 6 de septiembre de 2019

A mi familia y amigos, por su apoyo estos años

Agradecimientos

En primer lugar, quiero agradecer a mi familia, por su apoyo estos años y por los valores que me transmitieron desde siempre.
A mis amigos, por hacerme olvidar todos mis problemas durante unas horas todas las semanas.
Y finalmente, a mis tutores, Álex y Miguel, por implicarse tanto en la elaboración de este proyecto, mucho más allá de lo que les correspondía.

Resumen

El objetivo de este trabajo de fin de grado es desarrollar una aplicación web para gestionar información de valor para personas con movilidad reducida sobre las calles y destinos, usando esta información para realizar un cálculo de rutas personalizado en base a la movilidad del usuario, teniendo en consideración esta información. Esto permite producir una alternativa a los clásicos navegadores personalizada para personas con minusvalías.

Para alcanzar este objetivo fue necesario, en primer lugar, realizar un análisis previo sobre las funcionalidades y objetivos que abarcaría el proyecto, los requisitos funcionales que debe cumplir el producto final y la viabilidad de los mismos. A continuación se realizó el diseño, implementación y pruebas de las funcionalidades de la aplicación derivadas del análisis previo.

En el desarrollo se empleó el lenguaje Java junto con el framework Spring para producir un servicio REST que albergase lógica de negocio necesaria para el funcionamiento de la aplicación, el framework ORM Hibernate sobre un SGBD PostgreSQL para el almacenamiento de información, así como un cliente web implementado con el framework JavaScript Vue.js para la visualización.

El trabajo de fin de grado se gestionó siguiendo una metodología iterativa e incremental para el desarrollo de software.

Abstract

The goal of this end-of-degree project is to develop a web application to manage useful information for reduced mobility people about streets and destinations, using this information to perform a custom route calculation based on user's mobility, taking into account this information. This allows to produce a custom alternative to classic navigators for people with disabilities.

In order to achieve this objective, it was necessary, first of all, to carry out a preliminary analysis of the functionalities and objectives covered by the project, the functional requirements that the final product must meet and their viability. Next, the design, implementation and testing of the application functionalities derived from the previous analysis was carried out.

In the development, the Java language was used together with the Spring framework to produce a REST service that houses business logic necessary for the operation of the application, the Hibernate ORM framework on a PostgreSQL DBMS for information storage, as well as a web client implemented with the JavaScript framework Vue.js for visualization.

The final end-of-degree project was managed following an iterative and incremental methodology for software development.

Palabras chave:

- Java
- Accesibilidad
- Aplicación web
- Cálculo de rutas
- Navegación
- Sistema de información geográfica
- Spring
- Hibernate
- GeoJSON
- Vue
- Leaflet

Keywords:

- Java
- Accesibility
- Web application
- Route calculation
- Sailing
- Geographic information system
- Spring
- Hibernate
- GeoJSON
- Vue
- Leaflet

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En este capítulo se tratará la motivación que dio lugar a la elaboración del proyecto y los objetivos que se pretenden alcanzar en este.

1.1 Motivación

Aunque existen sistemas de información geográfica con cálculo de rutas y navegación, no existen muchas alternativas que soporten información de interés con sobre la accesibilidad de los diferentes destinos y calles y con una funcionalidad orientada a personas con movilidad reducida, y las existentes no permiten realizar cálculo de rutas y navegación teniendo en cuenta dicha información.

Debido a la creciente popularidad y funcionalidad de Internet, las aplicaciones web son una de las formas más comunes utilizadas para publicar información geográfica. Recientemente ha surgido una gran cantidad de bibliotecas que permiten la integración de la funcionalidad de visualización de información geográfica en una aplicación web.

Por lo tanto, el objetivo principal de este trabajo es crear una herramienta web que agregue a la gestión típica de la información geográfica, la posibilidad de gestionar información sobre los elementos, presentes en las calles y los destinos, que pueden limitar o facilitar la movilidad de las personas con algún tipo de discapacidad o lesión que les impide viajar a través de áreas normales. Para facilitar esto, también se integrará un cálculo de ruta personalizado basado en estos elementos, lo que permitirá a los usuarios dirigirse a través de las rutas en las que pueden circular.

1.2 Objetivos

La motivación expuesta al final de la sección 1.1 desencadena en los siguientes objetivos del proyecto:

- Visualizar la información geográfica sobre la accesibilidad de las distintas calles y destinos soportados por la aplicación en un mapa interactivo para el usuario. Estos datos provendrán de distintas fuentes de información, como Open Street Maps, Lídar, Redes sociales, usuarios, etc.
- Proporcionar un sistema de registro de usuarios, con el que estos se puedan dar de alta en el sistema y guardar sus características personales en cuanto a la movilidad, para ofrecer un cálculo de rutas ajustado a sus necesidades
- Permitir la notificación de incidencias por parte de los usuarios, con lo que enriquecer la información sobre la accesibilidad en la aplicación.
- Presentar las funcionalidades de administración (creación, lectura, actualización y borrado) de la información a través de una interfaz gráfica que permita interactuar de forma intuitiva con el mapa en el que se visualizará esta.
- Disponer de sistema de roles de usuario, que diferencie entre usuarios clientes y usuarios administradores, que serán los encargados de manejar la información contenida en el sistema y la proporcionada por los usuarios clientes a través de las incidencias, que serán los consumidores principales del producto y que no tendrán acceso a las funcionalidades de gestión de la información.
- Desarrollar un algoritmo de navegación personalizado capacitado para dirigir a los usuarios en función de su limitación en la movilidad y de la información sobre la accesibilidad, esquivando zonas que dificulten o impidan su paso y destinándolos a plazas de aparcamiento adaptadas y entradas accesibles de su destino.
- Administrar información sobre destinos que se correspondan con lugares de interés, con información propia sobre las entradas accesibles de las que dispongan y que permitan interactuar de forma sencilla con ellos a los usuarios.

Capítulo 2

Fundamentos tecnológicos

2.1 Estado del arte

En esta sección se describen las herramientas presentes en el mercado con funciones similares a las de este proyecto y las tecnologías más relevantes que utiliza el mismo.

A partir de los objetivos expuestos en la Sección 1.2, se ha realizado una búsqueda en el mercado de herramientas que realicen funciones similares, con el fin de servir de fuente de información e inspiración para algunos aspectos de este proyecto y completar la funcionalidad del mismo:

- Google Maps [1]: en lo relativo a funciones de accesibilidad, Google Maps proporciona información sobre transportes públicos que disponen de accesos especiales para minusválidos, indicando los horarios de paso por cada una de las paradas (figura 2.1).

- Mapp4All [2]: se trata de una aplicación móvil que contiene información sobre la adaptabilidad de los distintos locales registrados en la herramienta para personas de movilidad reducida. Los usuarios pueden registrar este tipo de información sobre cada local directamente en la aplicación, a través de un mapa interactivo. Además, Mapp4All permite buscar locales filtrando solo aquellos que dispongan de determinados servicios adaptados a minusválidos (figura 2.2).

- Disabled Park [3]: aplicación disponible tanto en web como en Android contenedora de información sobre plazas de aparcamiento disponibles en distintas ciudades del territorio español, mostrándolas en un mapa con un enlace a google maps en cada una de las plazas. Estas funciones que presenta Disabled Park, están disponibles en muchas otras herramientas del mercado para zonas geográficas más concretas, como el sitio web discapacidadsevilla.org (figura 2.3).

- Accesibility Plus [4]: aplicación nativa de Android similar a Mapp4All en cuanto al

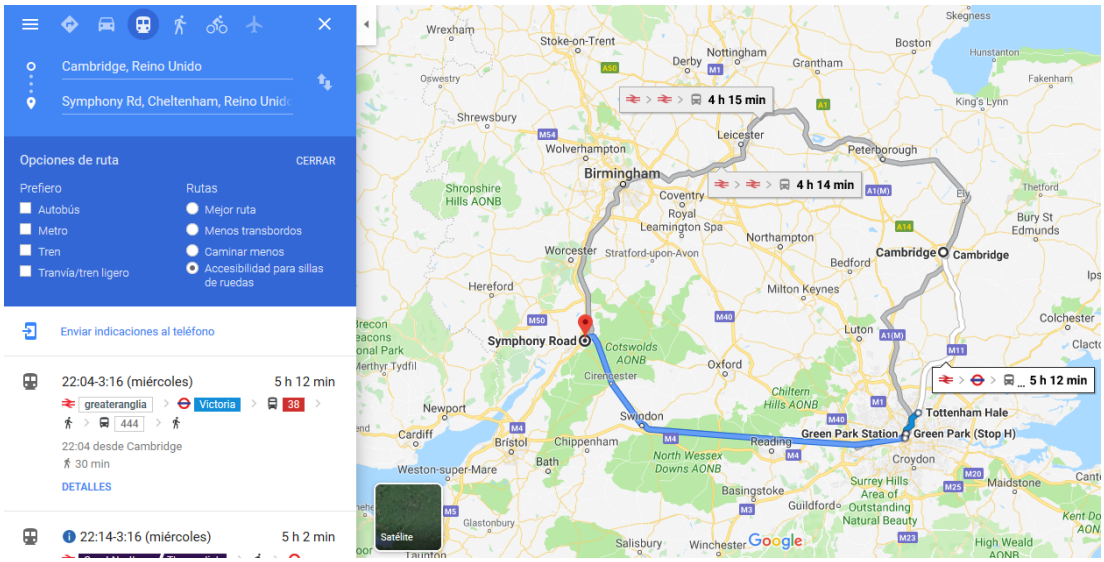


Figura 2.1: Opción de accesibilidad de Google Maps

Figura 2.2: Información de un punto de interés en Mapp4All

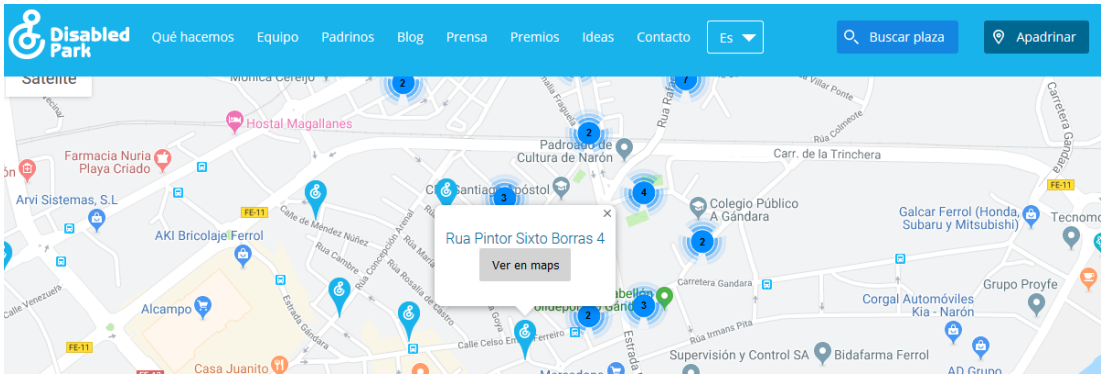


Figura 2.3: Mapa de Disabled Park

manejo de información sobre puntos de interés, pero que incorpora un sistema de notificaciones sobre actualizaciones en los datos para el tipo de información que configure el usuario (figura 2.4).

Todo este conjunto de herramientas presentan en común un aspecto: funcionan como contenedoras de información relativa a la accesibilidad, bien sea de locales o de transportes públicos. Sin embargo, ninguna presenta un cálculo de rutas específico (únicamente Google Maps se aproxima ligeramente a esta función, al dirigir que línea de transporte público escoger en función de la minusvalía del usuario). Además, sólo Google Maps permite guardar información relativa a la movilidad del usuario, para emplear esta en ofrecerle una herramienta personalizada a su discapacidad. Por último, al contrario que en este proyecto, ninguna maneja información sobre la accesibilidad de calles y caminos.

2.2 Tecnologías utilizadas

Las tecnologías más relevantes empleadas en el desarrollo del proyecto son:

- PostgreSQL [5]. Sistema gestor de base de datos que presenta las extensiones PostGIS [6] para el manejo de información geográfica y pgRouting [7] el cálculo de rutas.
- Hibernate [8]. Framework ORM (mapeo objeto/relacional) para automatizar la persistencia de los objetos de la aplicación en una base de datos relacional. La extensión Hibernate Spatial [9] da soporte a la persistencia de objetos con información geográfica.
- Spring Boot [10]. Spring es un framework para el desarrollo de aplicaciones de código abierto para la plataforma Java. Spring Boot facilita el desarrollo de proyectos basados en Spring.
- GeoJSON [11]. Formato para el intercambio de información geográfica basado en JSON.

Figura 2.4: Punto de interés en Accesibility Plus

- JUnit [12]. Conjunto de librerías utilizadas para hacer pruebas unitarias en Java.
- OSM2PO [13]. Herramienta que sirve como conversor de rutas de Open Street Map y motor de enrrutamiento.
- Leaflet [14]. Librería JavaScript para desarrollar mapas interactivos.
- Vue.js [15].Framework progresivo para construir interfaces de usuario. Su núcleo es bastante pequeño y se escala a través de plugins. Engloba en un mismo archivo HTML, CSS y JavaScript.
- Node.js [16].Plataforma de desarrollo para la creación de aplicaciones destinadas a la Web, orientada a redes y centrada en la velocidad y la escalabilidad.
- Bootstrap Vue [17]. Librería de estilos para elaborar aplicaciones web Responsive basadas en Vue.js.
- Axios [18]. Librería de promesas JavaScript que permite gestionar las llamadas AJAX realizadas desde el cliente web.

Capítulo 3

Metodología y planificación

En este capítulo se describe la metodología seleccionada para el desarrollo del proyecto y la adaptación al mismo, la planificación previa y el seguimiento de dicha planificación.

3.1 Metodología de desarrollo

La metodología seleccionada para el desarrollo es la metodología basada en iteraciones [19], donde cada iteración representa un conjunto de funcionalidades sobre las que realiza un ciclo software clásico (análisis, diseño, implementación y pruebas) y se añade al producto hasta conformar el producto final.

Se escoge esta metodología ya que permite reducir la complejidad de desarrollo del proyecto, al dividir el mismo en distintos bloques de funcionalidad. Además, permite desarrollar un producto de forma rápida, lo que facilita la retroalimentación en las reuniones de seguimiento del proyecto. Por otro lado, la implementación de una iteración proporciona experiencia al desarrollador para otras iteraciones, algo muy valioso en este contexto, ya que no existe trabajo previo de implementación con muchas de las tecnologías de las se vale este proyecto y muchas de ellas serán en cierto modo similares entre sí (iteraciones relacionadas con el manejo de la información en la aplicación). Por último, el producto final será más fiable, ya que al probar por separado cada iteración, estas se prueban exhasutivamente, por lo que la probabilidad de errores en el producto final es baja.

Se descarta el uso de otras metodologías como:

- Cascada [20]: en esta metodología se necesita un análisis previo perfecto del proyecto, ya que cualquier cambio en los requisitos del mismo implicaría repetir el proceso completo, lo cual puede disparar los costes del proyecto.

- Ágil [21]: las metodologías ágiles están orientadas a una interacción frecuente con el cliente. Esta situación no sucede en el contexto de este proyecto, por lo que se descarta

su uso.

- *Rational Unified Process* [22]: es una metodología de desarrollo incremental que se considera demasiado formal para este proyecto, ya que se cuenta con un único analista/programador, por lo que la adaptación de esta metodología al proyecto podría entorpecer el transcurso de este.

3.1.1 Adaptación de la metodología al proyecto

Como punto de partida del proyecto, se llevó a cabo un análisis preliminar, que incluyó:

- En primer lugar, el alcance del proyecto, para ver qué objetivos debía abarcar el proyecto (comentados en la Sección 1.2)
- También se buscaron herramientas con funciones similares en el mercado (detalladas en la Sección 2.1) que pudiesen servir de ayuda en el desarrollo de este proyecto.
- A continuación, se hizo un análisis de requisitos para determinar la funcionalidad que debe presentar el proyecto, representada mediante un modelo de casos de uso.
- En la parte análisis de requisitos, se elaboraron prototipos de pantalla (bocetos de la interfaz de usuario), en busca de nuevos casos de uso no detectados durante la realización del modelo anterior.
- Un modelo conceptual de datos, representado mediante un diagrama de clases, que indica como se estructura la información a almacenar en la aplicación, y la relación entre las diferentes entidades que la conforman..
- Modelo del servicio web REST, representado mediante un diagrama de clases que contiene la estructura de los recursos a manejar por parte del servidor y sus ubicaciones.

A continuación, se planificaron las tareas para llevar a cabo el proyecto y se comienza con el desarrollo iterativo del mismo, aplicando en cada iteración las fases de análisis, diseño, implementación y pruebas de las funcionalidades de cada iteración. Al final de cada iteración, se documentó el proceso realizado en ella, para lo que se usa esta memoria:

- Análisis: en la parte del análisis, se estudia exhaustivamente todos los requisitos y la funcionalidad que debe abarcar la iteración.
- Diseño: en el diseño se especifica una solución software para cumplir con los requisitos detectados durante la fase de análisis.
- Implementación: en esta fase se procede a la programación de la solución diseñada.

- Pruebas: se realizan pruebas unitarias sobre los artefactos creados y/o modificados, y pruebas globales sobre el funcionamiento del producto final de cada iteración.

3.1.2 Herramientas de apoyo a la metodología

Para aplicar esta metodología, se utilizaron una serie de herramientas para facilitar la adaptación de esta al proyecto:

- Dia [23]. Software libre utilizado para la elaboración de los diagramas de casos de uso y de clases.
- Balsamiq Mockups [24]. Programa de elaboración de prototipos de pantalla.
- Maven [25]. Herramienta de gestión de y construcción de proyectos Java.
- Eclipse [26]. IDE de programación utilizado para la parte servidor en Java.
- Npm [27]. Gestor de paquetes de JavaScript.
- Sublime [28]. Editor de texto utilizado para la implementación de la parte cliente en Vue.js.
- GitLab [29]. Herramienta de control de versiones del software producido.
- Notepad [30]. Editor de texto con el que se desarrollan los scripts SQL.
- Postman [31]. Herramienta para el testeo del API REST del servidor web.

3.2 Planificación y seguimiento

En esta sección se describe la planificación inicial de las tareas para la consecución del proyecto, los recursos humanos y técnicos disponibles y las desviaciones surgidas durante el seguimiento de la mencionada planificación inicial.

3.2.1 Recursos

Recursos Humanos

Los recursos humanos con los que cuenta este proyecto están compuestos por 3 personas: dos jefes de proyecto (los directores de este proyecto) y un analista/programador. Los jefes de proyecto desempeñan las funciones de supervisión del proyecto, así como de análisis y diseño del proyecto, mientras que el estudiante analista y programador se encarga de la planificación, análisis, diseño, implementación y pruebas de cada una de las iteraciones que conforman el proyecto.

Recursos Técnicos

Los recursos hardware con los que cuenta este proyecto se componen únicamente del ordenador personal del alumno Asus X555U.

En cuanto a los recursos software, estos se detallaron en las secciones Sección 2.2 y Sección 3.1.2, suponiendo todos ellos softwares o librerías de acceso gratuito, por lo que el uso de estos recursos no supone ningún coste adicional al proyecto. La herramienta de control de versiones GitLab fue proporcionada por el grupo de investigación del Laboratorio de Bases de Datos, al cual pertenecen los directores del proyecto.

3.2.2 Planificación inicial

El proyecto se compone de 4 iteraciones, que engloban bloques funcionales directamente relacionados cada una de ellas:

- Primera iteración: en la primera iteración se realizan la tareas relacionadas con el registro de usuarios en el sistema, la distinción de roles entre y el tratamiento de la información de los elementos de accesibisibilidad de las calles. Se estima una duración de cuatro semanas para el desarrollo completo de esta versión del producto.
- Segunda iteración: en esta iteración se llevan a cabo todas las tareas relacionadas con el cálculo de rutas personalizado la movilidad del usuario que acciona el caso de uso. Se estima una duración de tres semanas para el desarrollo completo de esta versión del producto.
- Tercera iteración: en esta iteración se llevan a cabo el trabajo relacionado el manejo de la información de los destinos que se visualizarán en la aplicación. Se estima una duración de tres semanas para el desarrollo completo de esta versión del producto.
- Cuarta iteración: en esta última iteración se realiza el trabajo relacionado con la gestión de las incidencias que reportarán los usuarios, y que serán atendidas por los administradores.

Conjuntamente al trabajo en cada una de las iteraciones mencionadas, se irá redactando la documentación del proyecto, contenida en esta memoria. Finalmente, se dejará un periodo de tres semanas para la redacción y corrección de esta memoria.

La planificación inicial del proyecto es representada en el diagrama de Gantt de la figura 3.1, teniendo como fecha de inicio el día 1 de marzo de 2019 y fecha de fin el día 21 de junio de 2019

En lo relativo a los costes del proyecto, se toma una media de trabajo de 4 horas diarias dedicadas por el analista/programador. El coste en tiempo de los jefes de proyecto se mide

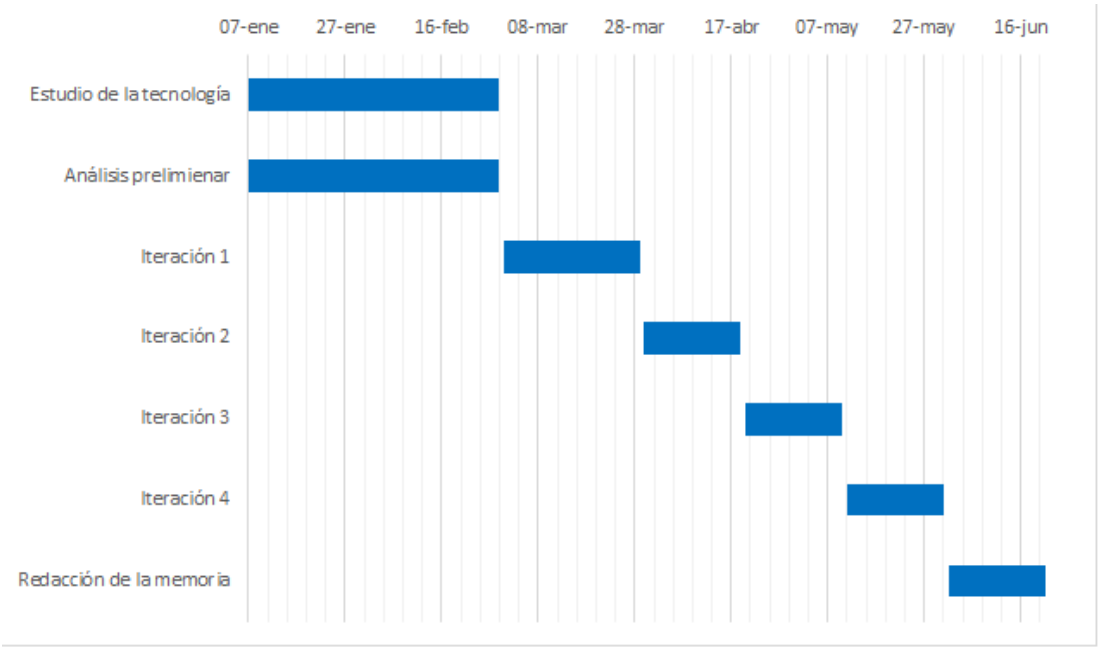


Figura 3.1: Diagrama de Gantt de la planificación inicial

sumando las reuniones de seguimiento del proyecto, con una media de una hora por reunión, con una periodicidad de dos semanas.

El coste de euro por hora se calcula con en base al salario bruto del trabajador, añadiendole el coste de Seguridad Social y dividiéndolo entre las horas laborables del año:

$Coste = \frac{salario \times 1{,}33}{1500}$

Tomando como salario bruto del analista/programador 24.000€ y de los jefes de proyecto de 36.000€, dichos cálculos resultan, redondeando, en:

Analista/programador $= \frac{24000 \times 1{,}33}{1500} \approx 21€/h$

Jefe de proyecto $= \frac{36000 \times 1{,}33}{1500} \approx 31€/h$

Teniendo en cuenta estos costes y los tiempos planificados para el proyecto, se obtiene la tabla resumen 3.1 con los costes en horas y en euros de cada uno de los recursos humanos.

Recurso	Horas	Coste
Analista/programador	456 h	9.576€
Jefe de proyecto	12 h	372€
Jefe de proyecto	12 h	372€
Total	570 h	10.320€

Cuadro 3.1: Coste en horas y en euros de los recursos humanos

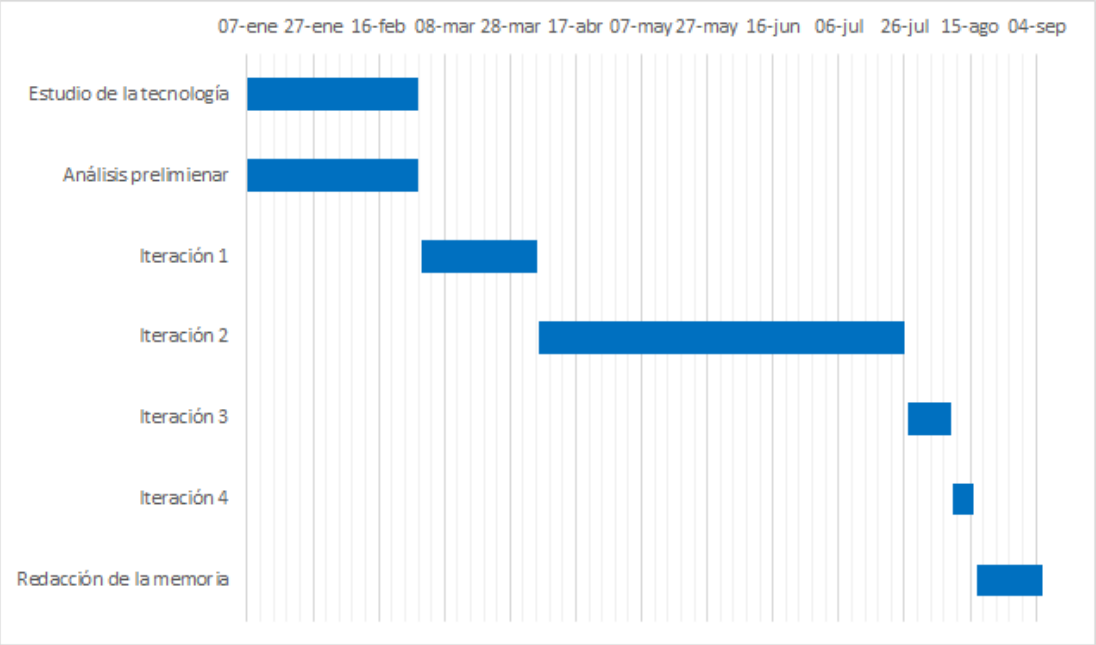


Figura 3.2: Diagrama de Gantt de la replanificación del proyecto

3.2.3 Seguimiento

Durante el seguimiento de la planificación detallada en la sección 3.2.2 se han detectado dos desviaciones significativas en el tiempo estimado para la ejecución de las tareas del proyecto:

- La primera iteración duró una semana más de lo previsto, debido a las dificultades encontradas en el desarrollo de esta, al tratarse de nuevas tecnologías con las que el desarrollador no había trabajado previamente.
- En la segunda iteración se produce la desviación más relevante y que obliga a realizar una replanificación del proyecto: el desarrollo del algoritmo de cálculo de rutas se complica y, al llevar tres semanas de trabajo en esta iteración sin un progreso significativo, se decide replanificar el proyecto nuevamente.

Tras estas desviaciones se realiza una nueva planificación del proyecto a fecha fin el día 6 de septiembre de 2019, coincidiendo con la entrega de esta memoria. Viendo los problemas encontrado en el desarrollo de la segunda iteración, se decide poner la mayoría de las horas disponibles en esta iteración, dejando el diagrama de Gantt de la figura 3.2 sobre la nueva planificación del proyecto.

Asimismo, se recalculan los costes de los recursos involucrados en el proyecto, plasmados en la tabla resumen 3.2.

Esta segunda planificación del proyecto si se cumple exitosamente. Tras la experiencia previa, se decidió dedicarle el mayor número de horas a la segunda iteración, al ver la complejidad y criticidad de la misma. Se acortaron los tiempos de las tercera y cuarta iteración, al ver que el trabajo realizado en la primera serviría como base sólida para afrontarlas.

Recurso	Horas	Coste
Analista/programador	676 h	14.196€
Jefe de proyecto	17 h	527€
Jefe de proyecto	17 h	527€
Total	710 h	15.250€

Cuadro 3.2: Tabla resumen con los costes de cada recurso tras la replanificación realizada

Capítulo 4

Análisis

4.1 Requisitos

En esta sección se describen los actores que interactuarán con el sistema, se enumeran los requisitos del proyecto y se detallan los requisitos funcionales y no funcionales que surgen de los requisitos del proyecto.

4.1.1 Actores

En la figura 4.1 se muestran los actores [32] que representan los roles que poseen los distintos usuarios que interactúan con el sistema.

En la aplicación se contemplarán 3 actores, como se puede ver en la Figura 4.1:

- Usuario no autenticado. Los usuarios no autenticados tendrán acceso a las acciones que no requieran de un rol de administrador ni a las acciones gestión de datos relativos a usuarios registrados en el sistema.
- Cliente. Los clientes interectuarán con los casos de usos a los que tiene acceso el usuario no autenticado y a todos aquellos que se correspondan con la gestión de la información de los clientes. Serán los consumidores principales del producto software surgido de la consecución del proyecto.
- Administrador. Los administradores interectuarán con los casos de usos a los que tiene acceso el usuario no autenticado y a todos aquellos que se corresponden con la gestión de la información de los elementos, destinos e incidencias del sistema.

4.1.2 Requisitos generales del proyecto

Los requisitos [33] que debe cumplir este proyecto son:

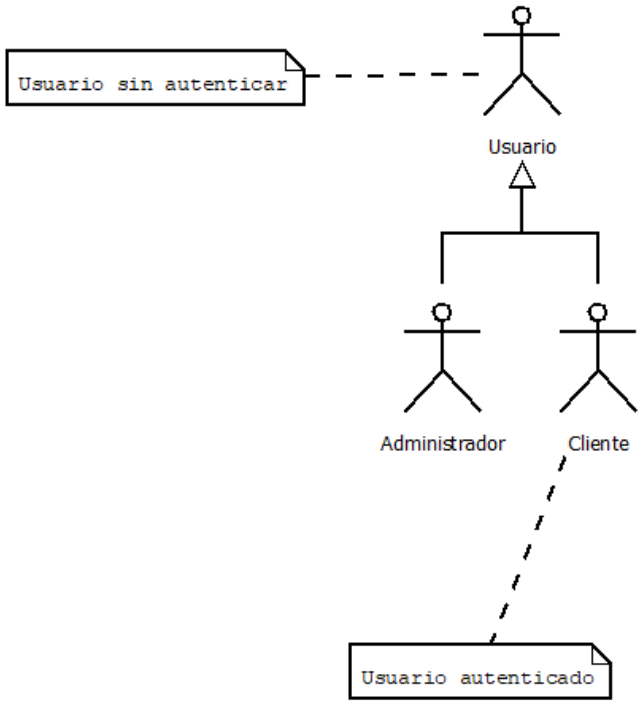


Figura 4.1: Jerarquía de actores del sistema

- Registro y autenticación de usuarios: un usuario podrá darse de alta en el sistema y acceder a este con las credenciales proporcionadas.
- Modificación del perfil de usuario: un usuario auteticado podrá modificar sus datos desde un perfil de usuario.
- Representación visual de los elementos en un mapa: los elementos registrados en la aplicación se podrán visualizar y consultar en un mapa interactivo.
- Gestión de la información de los elementos: se podrán crear, borrar y actualizar elementos en la aplicación.
- Cálculo de rutas personalizado a personas con movilidad reducida: la aplicación soportará un cálculo de rutas adaptado a la capacidad de movilidad del usuario.
- Representación visual de las incidencias en un mapa: las incidencias notificadas en la aplicación se podrán visualizar y consultar en un mapa interactivo.
- Gestión de la información de las incidencias: se podrán crear, borrar y actualizar incidencias en la aplicación

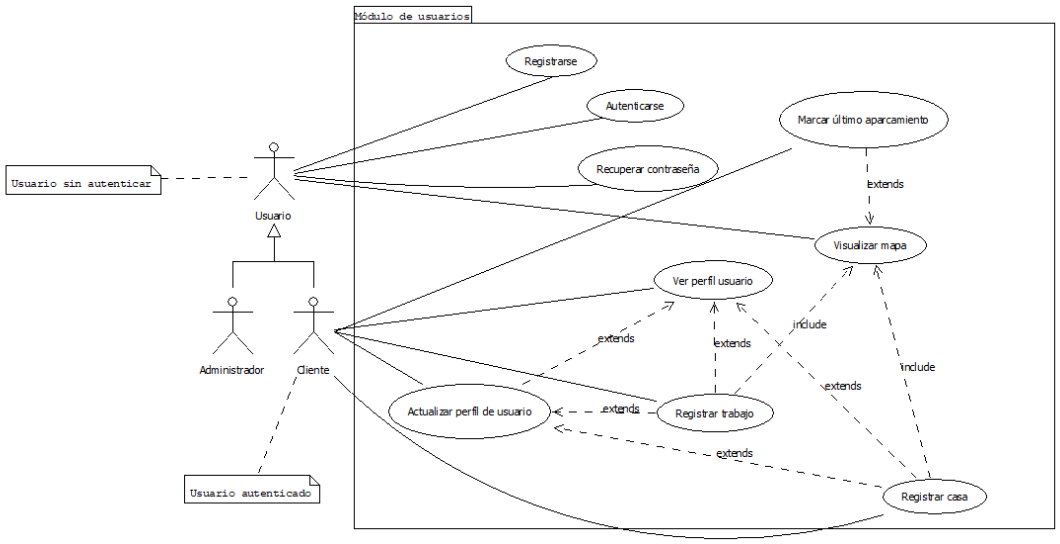


Figura 4.2: Diagrama de casos de uso del módulo de usuarios

- Representación visual de los destinos en un mapa: los destinos almacenados en el sistema se podrán visualizar y consultar en un mapa interactivo.

- Gestión de la información de los destinos: se podrán crear, borrar y actualizar destinos en la aplicación

4.1.3 Requisitos funcionales

Los requisitos funcionales de este proyecto se representan mediante un modelo de casos de uso [34]. Este modelo consta de un diagrama de casos de uso por cada módulo funcional del proyecto.

Módulo de usuarios

Este módulo abarca la funcionalidad relacionada con el registro y autenticación de los usuarios en la aplicación y la gestión de la información relativa a ellos.

A continuación se detallan los casos de uso que se muestran en el diagrama de la Figura 4.2:

- Registrarse: un usuario no autenticado puede darse de alta en el sistema como usuario cliente.

- Autenticarse: un usuario no autenticado puede acceder al sistema como usuario registrado con las credenciales asociadas al mismo.

- Recuperar contraseña: un usuario no autenticado puede recuperar la contraseña de un usuario, introduciendo la dirección de correo electrónico asociada a este, a la que se enviará un correo con la nueva contraseña.

- Ver perfil de usuario: un usuario cliente podrá ver los datos asociados al mismo que están guardados en el sistema.

- Actualizar perfil de usuario: un usuario cliente podrá modificar sus datos en la aplicación desde el perfil de usuario.

- Visualizar mapa: se podrá visualizar un mapa que contendrá toda la información geográfica registrada en el sistema (elementos, incidencias y destinos) y con el que se podrá interactuar para acceder a información más detallada, realizar otras acciones, etc.

- Registrar trabajo: en la actualización del perfil de usuario se puede registrar la ubicación del trabajo, lo que implica visualizar un mapa.

- Registrar casa: en la actualización del perfil de usuario se puede registrar la ubicación de la casa, lo que implica visualizar un mapa.

- Marcar último aparcamiento: desde el mapa principal se podrá seleccionar el punto correspondiente al último estacionamiento del usuario cliente.

Módulo de elementos

Este módulo da soporte a la funcionalidad relacionada con la gestión y visualización de los elementos limitadores o facilitadores de la movilidad registrados en la aplicación.

A continuación se detallan los casos de uso que se muestran en el diagrama de la Figura 4.3:

- Crear elemento: desde el mapa, un usuario administrador puede seleccionar la ubicación y cubrir los datos del nuevo elemento a guardar en la aplicación.

- Elemento detalle: si un usuario anónimo selecciona un elemento del mapa se verán toda la información relativa a este.

- Actualizar elemento: desde el detalle del elemento, un administrador puede elegir actualizar la información de este, modificando los datos del elemento en un formulario.

- Borrar elemento: desde el detalle del elemento, un administrador puede elegir borrar el elemento de la aplicación.

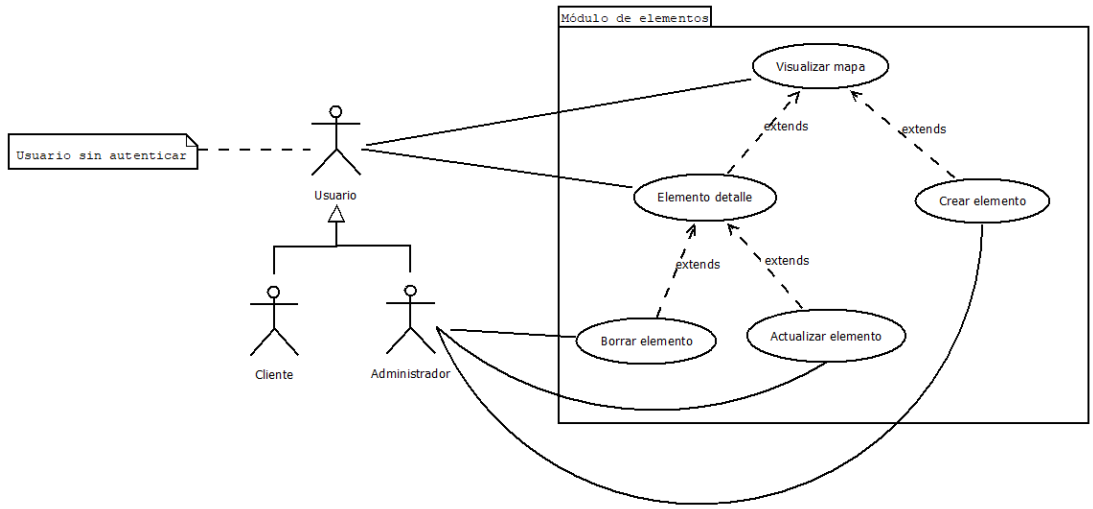


Figura 4.3: Diagrama de casos de uso del módulo de elementos

Módulo de cálculo de rutas

Este módulo abarca la funcionalidad relacionada con el cálculo de rutas personalizado y las funcionalidades de navegación de la aplicación.

A continuación se detallan los casos de uso que se muestran en el diagrama de la Figura 4.4:

- Buscar dirección: desde el mapa, un usuario anónimo podrá buscar pueblos o ciudades por su nombre.
- Calcular ruta punto a punto: un usuario anónimo puede seleccionar el cálculo de la ruta más cercana, esquivando todos los elementos que le impidan el paso según la movilidad indicada, de un punto del mapa hasta otro, lo que involucrará seleccionar esos puntos origen y destino. Se calculará la ruta pasando por el último aparcamiento (en el caso de ser un usuario cliente) y la plaza adaptada más cercana al destino.
- Marcar origen: desde el mapa, un usuario anónimo podrá seleccionar un punto como origen de la ruta a calcular.
- Marcar destino: desde el mapa, un usuario anónimo podrá seleccionar un punto como destino de la ruta a calcular.
- Empezar navegación: una vez calculada la ruta, un usuario anónimo podrá comenzar la navegación por la misma, viendo su posición en el mapa.
- Marcar último aparcamiento: un usuario cliente podrá indicar, bien directamente desde el mapa seleccionando un punto, o bien durante la navegación por la ruta (que sería la posición actual del usuario), la ubicación donde aparcó por última vez su vehículo.

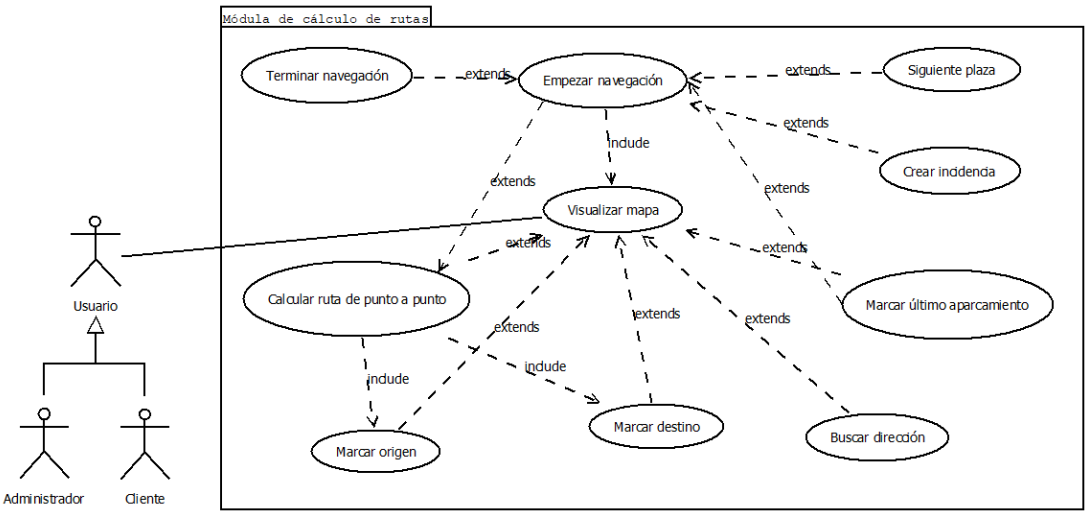


Figura 4.4: Diagrama de casos de uso del módulo de cálculo de rutas

- Crear incidencia: tras comenzar la navegación, un usuario cliente podrá crear una incidencia directamente (en la posición actual del usuario) en el sistema.
- Siguiente Plaza: un usuario anónimo podrá seleccionar el recálculo de la ruta para que esta pase por la siguiente plaza de aparcamiento más cercana al destino en un radio de un 1 kilómetro.
- Terminar navegación: un usuario anónimo podrá elegir terminar el proceso de navegación.

Módulo de destinos

Este módulo abarca la funcionalidad relacionada con la gestión de la información de los destinos de la aplicación.

A continuación se detallan los casos de uso que se muestran en el diagrama de la Figura 4.5:

- Ver desitnos usuario: desde su perfil de usuario, un cliente podrá ver un listado con los destinos marcados como favoritos y sus datos más relevantes.
- Listar destinos: un usuario anónimo podrá ver en un listado todos los destinos presentes en la aplicación y sus datos más relevantes.
- Buscar destino: un usuario anónimo podrá buscar en el listado los destinos que coincidan con el nombre que introduzca.
- Ver destinos cercanos: un usuario anónimo podrá ver los destinos de cada tipo más cercanos a su posición.

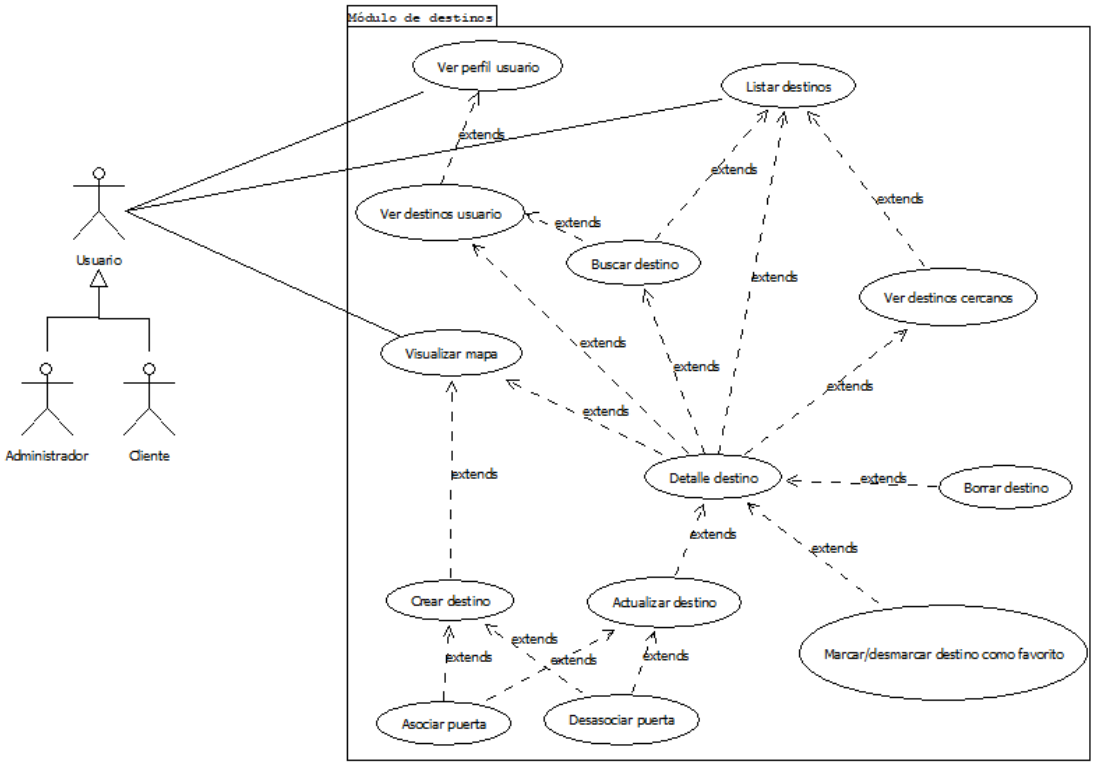


Figura 4.5: Diagrama de casos de uso del módulo de destinos

- Destino detalle: desde uno de los listados de destinos o tras seleccionar uno, un usuario anónimo podrá ver toda la información relativa a un destino existente en la aplicación.
- Borrar destino: desde el detalle de un destino, un administrador puede seleccionar borrar dicho destino de la aplicación.
- Marcar/desmarcar destino como favorito: desde el detalle de un destino, un cliente puede marcar o desmarcar este como destino favorito
- Actualizar destino: desde el detalle de un destino, un administrador puede seleccionar modificar la información relativa al mismo, introduciendo los nuevos datos en un formulario.
- Crear destino: un administrador puede seleccionar en el mapa un punto que será la ubicación del destino a dar de alta, del que deberá rellenar los datos en un formulario.
- Asociar puerta: un administrador podrá asociar una puerta de accesibilidad del destino desde el formulario de crear o modificar destino.
- Desasociar puerta: un administrador podrá desasociar una puerta de accesibilidad del destino desde el formulario de crear o modificar destino.

Módulo de incidencias

Este módulo abarca la funcionalidad relacionada con la gestión de la información de la notificación de incidencias en la aplicación.

A continuación se detallan los casos de uso que se muestran en el diagrama de la Figura 4.6:

- Crear incidencia: desde el mapa, un cliente puede seleccionar el punto donde desee notificar una incidencias, de la que debe aportar un comentario descriptivo para guardarla en el sistema.
- Ver incidencias usuario: un cliente podrá visualizar las incidencias que ha notificado, y que aún no han sido atendidas polos administradores, directamente en el mapa.
- Ver incidencias pendientes: un administrador podrá visualizar las incidencias que han notificado todos los clientes, y que aún no han sido atendidas, directamente en el mapa.
- Listar incidencias: un administrador podrá ver el listado completo de incidencias registradas en la aplicación, con la información más relevante de estas.
- Detalle incidencia: un cliente o un administrador podrán ver toda la información relativa a una incidencia dada de alta por el cliente en la aplicación.

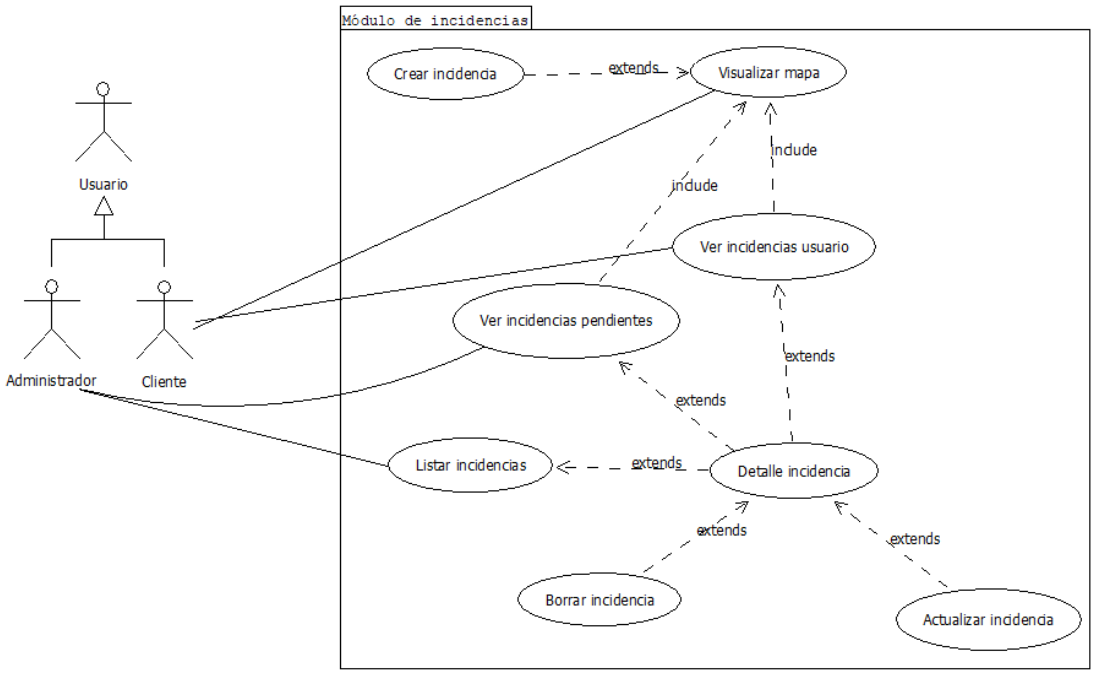


Figura 4.6: Diagrama de casos de uso del módulo de incidencias

- Actualizar incidencia: un administrador podrá actualizar una incidencia desde el detalle de esta.

- Borrar incidencia: un administrador podrá eliminar del sistema una incidencia desde el detalle de esta.

4.1.4 Requisitos no funcionales

Los requisitos no funcionales surgidos de los requisitos del proyecto que debe cumplir la aplicación son:

- La aplicación debe poseer una interfaz de fácil uso e intuitiva para que cualquier pefil de usuario pueda utilizar todas las funcionalidades que ofrece esta.

- La interfaz debe ser adaptable a todo tipo de dispositivos sin que se pierda ninguna funcionalidad de la aplicación.

- Se debe diseñar una aplicación web para llegar al mayor número de usuarios potenciales posible.

4.2 Arquitectura del sistema

La arquitectura del sistema se basa en una arquitectura en tres capas [35] (figura 4.7), en el que se aísla la responsabilidad de cada capa mediante el patrón fachada. Cada una se comunicará únicamente con la capa inmediatamente inferior. Este patrón de arquitectura otorga una gran escalabilidad y flexibiildad al sistema, ya que cambios en capas inferiores no afectan a las superiores.

Las capas de la arquitectura del sistema son:

- La capa modelo será la encargada de encapsular toda la funcionalidad de acceso a datos, comunicándoselos a la capa controlador. Abstraerá a las demás capas de la comunicación SQL con el sistema gestor de base de datos relacional.
- La capa controlador se compondrá de un servidor de aplicaciones en el que se encapsulará la lógica de negocio del servidor, utilizando para el acceso a datos la capa modelo. Tendrá una interfaz REST para la comunicación de la vista.
- La capa vista se compondrá de un servidor web que realizará llamadas a la interfaz REST de la capa controlador, para ofrecer toda la funcionalidad que requiere el proyecto mediante una interfaz de usuario.

4.3 Interfaz de usuario

En cuanto a la interfaz de usuario, esta se compondrá de una barra superior con enlaces a las distintas pantallas de la aplicación y del contenido de la pantalla actual. Existirá un pantalla principal, compuesta por un mapa en el que se visualizarán todos los elementos, destinos e incidencias (estas últimas dependiedo del rol del usuario) de la aplicación, y en la que se desplegará una barra lateral para acceder a las acciones que requieran interacción con el mapa. Además existirán pantallas indivudales con listados e información del usuario.

Los componentes de los que constará la interfaz de usuario de esta aplicación son:

- La pantalla principal, que será el nucleo de toda la interfaz. Desde esta pantalla se visualizarán los elementos, destino e incidencias de la aplicación en sus respectivas ubicaciones, se realizarán las acciones que requieran de interacción con el mapa a través de un menú de opciones en el propio mapa: seleccionar los puntos origen y destino de la ruta a calcular, marcar el último aparcamiento del usuario y navegar a los formularios de creación de elementos, destinos e incidencias dependiendo del rol usuario, con la ubicación de estas en el punto seleccionado en el mapa. Se desplegará una barra lateral con el contenido de cada subpantalla que corresponda:

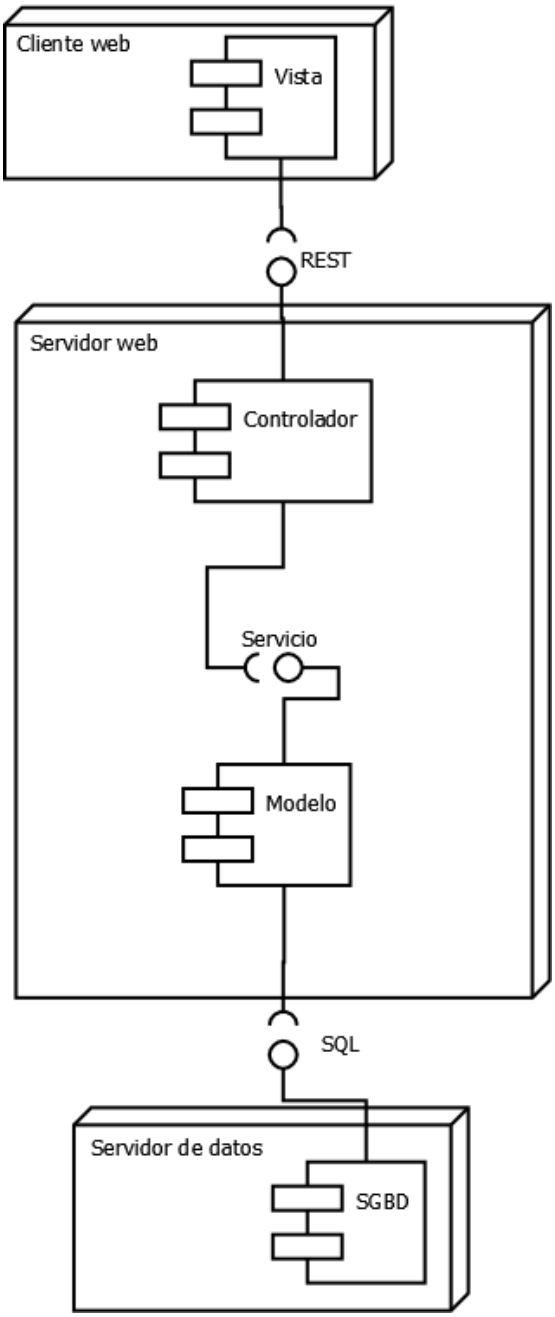


Figura 4.7: Diagrama de componentes de la arquitectura del sistema

  - Detalle del elemento. Al pinchar sobre un elemento, el contenido de la barra serán los datos del elemento clicado previamente en el mapa de la pantalla principal. Contendrá los botones eliminar y editar, que redigirán al usuario a la pantalla principal y al formulario del elemento respectivamente.
  - Formulario del elemento. Formulario con los campos necesarios para modificar los datos de cada elemento (en función de su clase), que al confirmar llevará a la subpantalla detalle elemento. Accesible también desde el menú de opciones de la pantalla principal, en la opción crear elemento.
  - Detalle destino. Al seleccionar un destino en el mapa, aparecerán los datos de dicho destino. Los botones editar y eliminar redigirán al formulario destino y a la pantalla principal respectivamente.
  - Formulario destino. Contendrá los campos necesarios para modificar los datos de un destino. Se puede acceder a el, además de desde el detalle destino, desde la opción crear destino del menú de la pantalla principal.
  - Detalle incidencia. Al seleccionar una incidencia en el mapa, se mostrarán los datos de esta en la barra lateral. Al igual que en los casos anteriores, los botones editar y eliminar dirigirán al usuario al formulario y a la pantalla principal respectivamente.
  - Formulario incidencia. Subpantalla desde la que se podrán modificar los datos de una incidencia. Se podrá navegar a ella también desde la opción crear incidencia del menú de opciones.
  - Cálculo de rutas. Formulario en el que se rellanarán los datos necesarios para realizar un cálculo de rutas, pintando la ruta calculada en el mapa de la pantalla principal. Se podrá acceder a el al seleccionar un punto origen o un punto destino en el menú de opciones del mapa.

  Al realizar el cálculo de ruta, aparecerá en la pantalla principal un botón para iniciar la navegación, lo que provocará que aparezcan los botones necesarios para acceder a todas las funcionalidades de esta: marcar aparcamiento, buscar siguiente plaza de aparcamiento y crear destino. La figura 4.8c contiene los prototipos de pantallas de todo el proceso desde el cálculo de la ruta hasta la navegación por la misma.

- Formulario de registro. En esta pantalla, accesible desde la barra de navegación de la aplicación, un usuario podrá registrarse como cliente en la aplicación.
- Formulario de autenticación. Pantalla accesible desde la barra de navegación en la que el usuario podrá autenticarse en la aplicación, ya sea como cliente o como administrador. Existirá un enlace que diriga al formulario de recuperación de contraseña

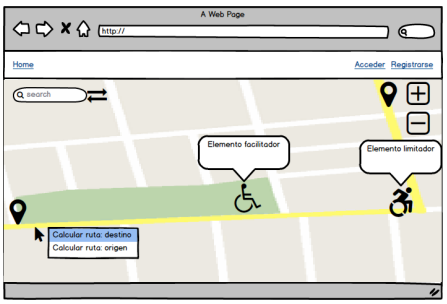
- Formulario de recuperación de contraseña. Desde este formulario, un usuario, bien sea cliente o bien sea administrador, podrá solicitar que se le envíe una contraseña nueva al correo electrónico introducido (siempre y cuando esté exista en la base de datos).
- Perfil de usuario. Pantalla que contiene toda la información relativa a un usuario cliente y desde la que se podrá navegar al formulario de usuario.
- Formulario de usuario. En este formulario se pueden modificar los datos del usuario cliente en cuestión. Al confirmar los cambios, se regresará al perfil de usuario.
- Listado de destinos. Accesible desde la barra de navegación de la aplicación, contiene un tabla con todos los destinos de la aplicación y la información más relevante de los mismos, desde la que se podrá acceder al detalle de cada destino. Además existirá un enlace a la pantalla de destinos próximos.
- Destinos próximos. Conjunto de desplegables (uno por cada tipo de destino) con los destinos más próximos a la ubicación actual del usuario, a los cuales se podrá navegar al pinchar sobre ellos.
- Listado de incidencias. Lista de incidencias a la que se puede navegar desde la barra superior, que estará formada por una tabla que contendrá la información más relevante acerca de la incidencia y desde la que se podrá acceder a la incidencia en cuestión.

Se han realizado prototipos de pantalla, además de para la navegación, para todas las pantallas detalladas en esta sección. Por motivos de espacio, se omiten en el contenido de este documento y se incluyen todas estas en el anexo.

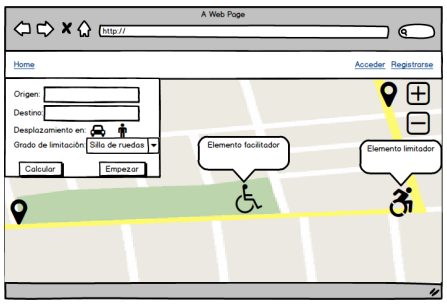
4.4 Modelo conceptual de datos

El modelo de datos de la aplicación estará formado por las siguientes entidades:

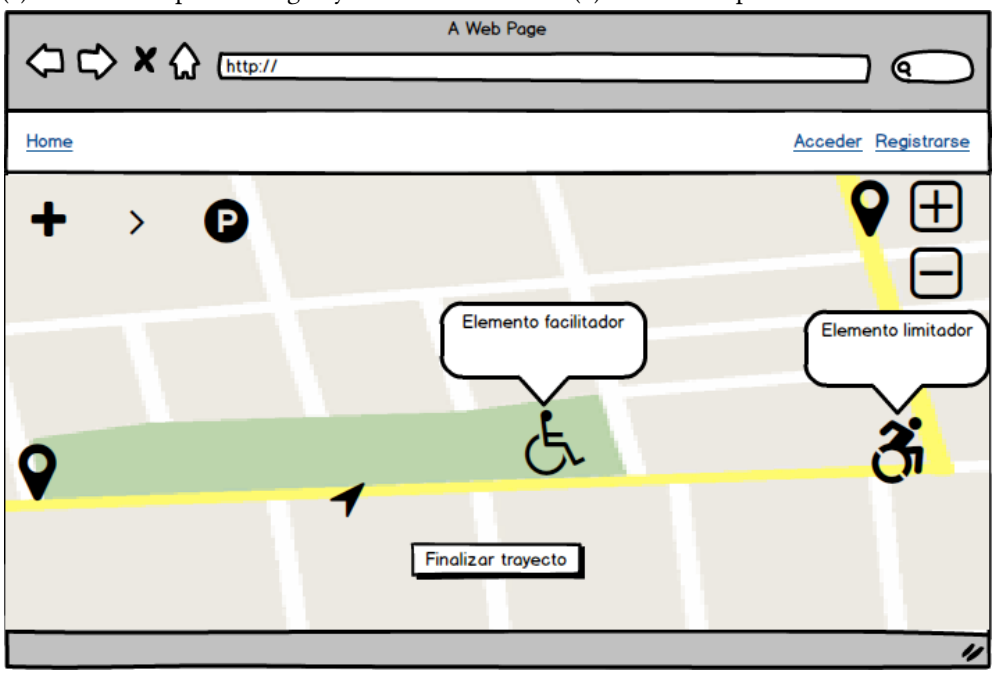
- Elemento: esta entidad representa a cualquier elemento relacionado con la accesibilidad de las calles y locales. Se especializa a su vez, en tres subclases:
  - Elemento limitador: obstáculo o impedimento para el acceso de una persona con movilidad reducida en una determinada ubicación.
  - Plaza de aparcamiento: plaza adaptada y reservada a personas de movilidad reducida.
  - Puerta: acceso de entrada a un local, edificio o zona.



(a) Selección de puntos origen y destino



(b) Formulario para el cálculo de rutas



(c) Navegación

Figura 4.8: Cálculo de ruta y posterior navegación

- Usuario: entidad que representa a todo aquel usuario de la aplicación que está registrado en el sistema. Se especializa en dos subclases:
  - Cliente: representa al consumidor principal de la aplicación, del que se guardan sus datos para ofrecer funcionalidad más personalizada a sus características.
  - Administrador: esta entidad representa a los usuarios que se encargarán de la gestión de la información de la aplicación.
- Destino: esta clase simboliza todos aquellos lugares o edificios que puedan ser más visitados.
- Incidencia: entidad que representa las notificaciones de los usuarios sobre elementos que puedan existir o no existir en las calles. Estas serán atendidas por los administradores, que decidirán dar de alta, dar de baja o modificar los elementos según la incidencia.
- Tramo peatonal: clase que simboliza un recorrido en la red de zonas peatonales.
- Nodo peatonal: entidad que representa el punto inicio o fin de un tramo de la red peatonal y en el que se pueden unir dos o más tramos de la red peatonal.
- Tramo vehículo: entidad que representa un recorrido en la red de carreteras.
- Nodo vehículo: clase que simboliza el punto de inicio o fin de un tramo de la red de carreteras y en el que se pueden unir dos o más tramos de la red de carreteras.

Las relaciones entre las entidades mencionadas y sus especificaciones se representan en el diagrama de clases de la figura 4.9

4.5 Interfaz del servicio web

Al tratarse de una aplicación web, se debe definir un servicio web, que será a la que el cliente web realizará las llamadas pertinentes.

El servicio web se compondrá de los siguientes recursos:

- Cuenta: este recurso es el utilizado para el manejo del registro, autenticación y sesión de los usuarios.
- Cliente: utilizado para la lectura de clientes y la modificación de sus datos.
- Cliente destinos: el recurso que representa los destinos marcados como favoritos por cada cliente.

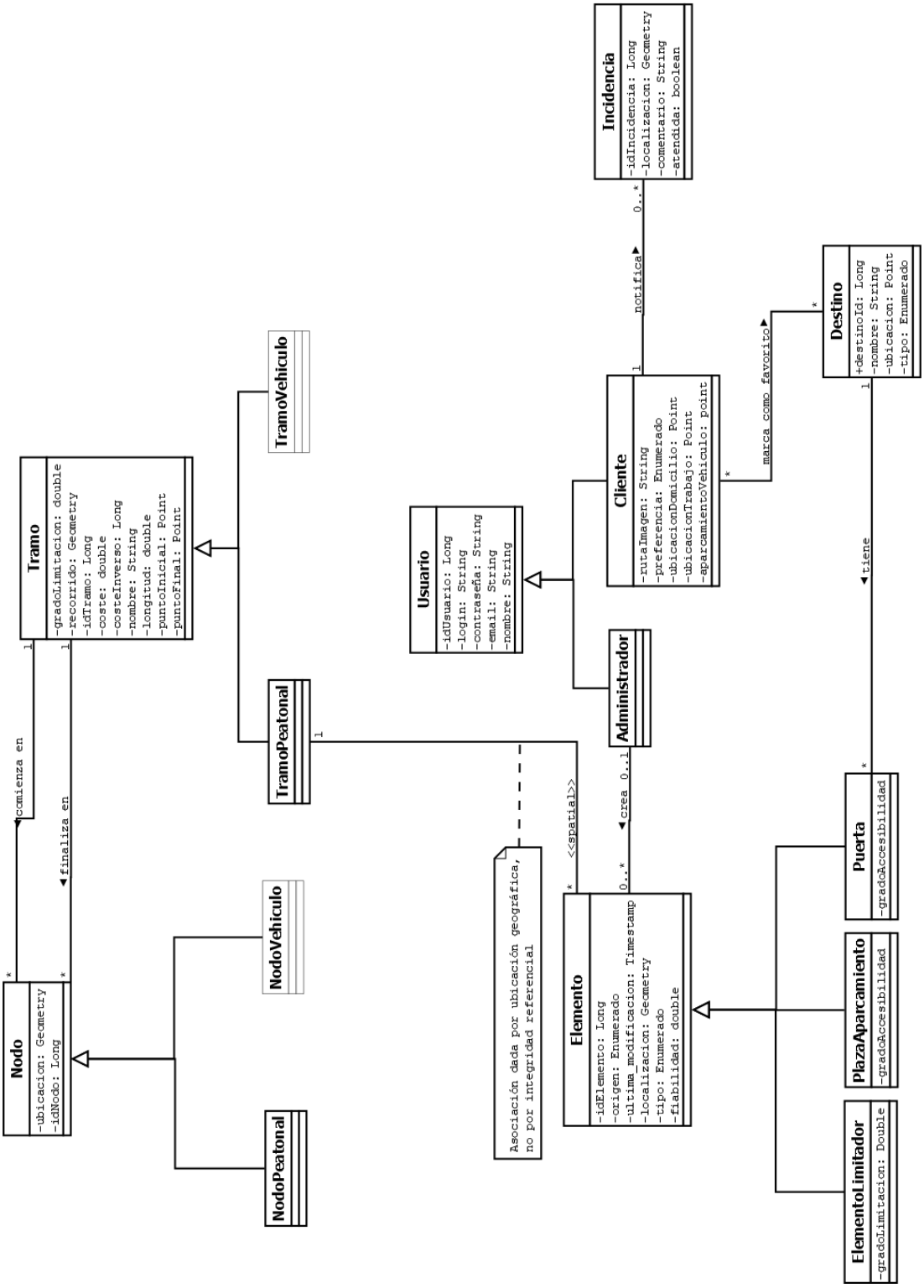


Figura 4.9: Diagrama de clases sobre el modelo conceptual de datos

- Cliente incidencias: recurso con las incidencias dadas de alta por cada cliente.
- Elemento: utilizado para la lectura, alta, modificado y borrado de todos los elementos del sistema.
- Plaza: recurso para la lectura de las plazas de aparcamiento.
- Limitador: recurso para la lectura de los elementos limitadores.
- Puerta: recurso para la lectura de las puertas.
- Incidencia: utilizado para la lectura, alta, actualización y borrado de las incidencias.
- Incidencia pendiente: recurso utilizado en la lectura de incidencias no atendidas por los administradores.
- Ruta: recurso para realizar el cálculo de rutas.
- Destino: recurso para la lectura, alta, modificación y borrado de los destinos.

El diagrama de la figura 4.10 muestra las rutas y métodos para acceder a cada recurso.

En dicho diagrama, cada clase representa a un recurso web y sus especificaciones. Cada flecha representa la navegación entre recursos, siendo el punto de partida el *front-controller* de Spring, y las ruta de acceso son el texto encima de las flechas que conducen desde el *front-controller* hasta el recurso. Los textos entre corchetes representan variables de la ruta.

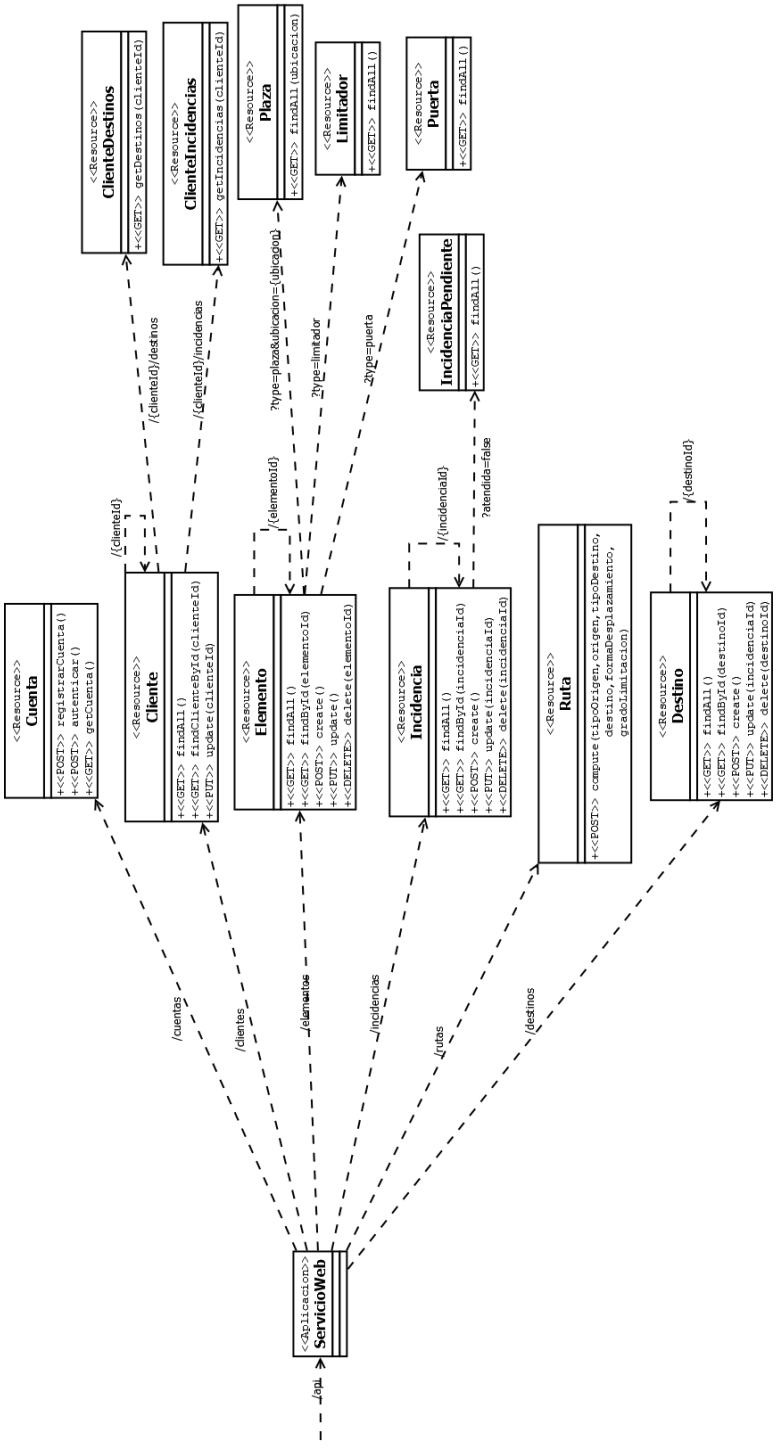


Figura 4.10: Diagrama de la interfaz del servicio web

Capítulo 5

Diseño

5.1 Arquitectura tecnológica del sistema

Partiendo de la arquitectura definida en la 4.2, en esta sección se describen las tecnologías utilizadas en cada uno de los componentes:

- Sistema gestor de base de datos: se utilizará el SGBD relacional PostgreSQL para el almacenamiento de los datos de la aplicación, realizándose la comunicación entre el modelo y el SGBD en el lenguaje SQL. Además se utilizarán las extensiones PostGIS, que da soporte a las operaciones y almacenamiento de información geográfica sobre un SGBD PostgreSQL, y pgRouting, que provee a PostGIS de la funcionalidad de cálculo de rutas.

- Acceso a datos: se realiza el acceso a la base de datos desde el servidor de aplicaciones utilizando el framework ORM Hibernate para la implementación de los DAOs, y la librería Spring Boot para la implementación de los servicios con los que se comunican los DAOs.

- Controlador: los controladores se implementan haciendo uso del framework Spring Boot, con el que se implementa un servicio REST al que se realizan llamadas desde el servidor web.

- Cliente web: El servidor web estará compuesto por un cliente web implementado con la librería Javascript Vue.js, haciendo uso de las librería Leaflet.js para la implementación de mapas en la vista y VueBootstrap para el diseño de la interfaz.

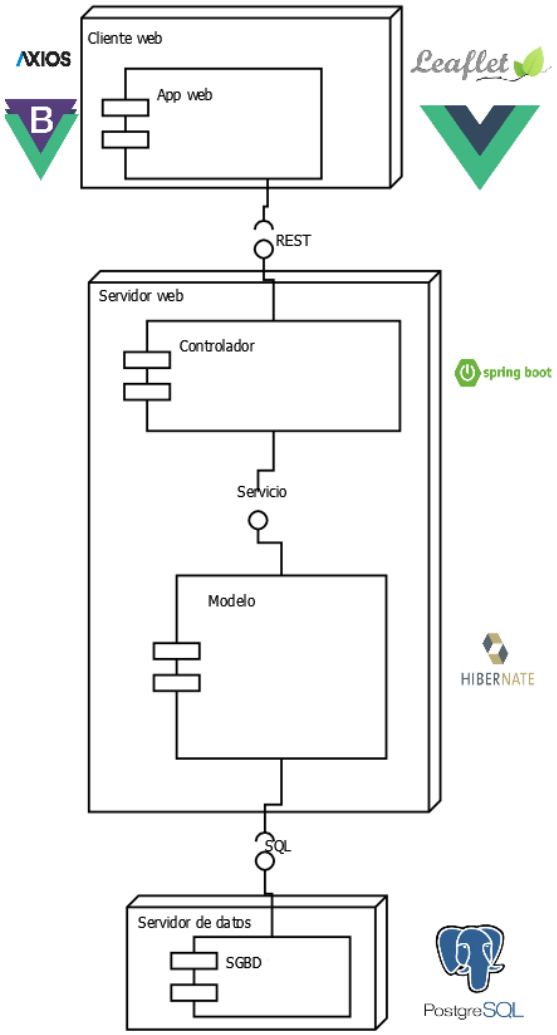


Figura 5.1: Diagrama de componentes de la arquitectura tecnológica del sistema

5.2 Diseño de la aplicación

En esta sección se detallan los componentes que conforman la arquitectura descrita en la Sección 5.1 y se describe la trazabilidad entre el análisis previo realizado en el capítulo 4 y la implementación de la aplicación.

5.2.1 Servidor

A continuación se describen los componentes más relevantes que conforman el *backend* de la aplicación.

Modelo

En el modelo se encuentran las clases que representan a las entidades de la aplicación, que serán los objetos que se persistirán en la base de datos.

Las especificaciones de dichas entidades y sus relaciones están descritas en el diagrama de clases de la Figura 4.9 en la Sección 4.4.

El código fuente de dichas entidades se encuentra en el paquete *es.udc.lbd.asi.servidortfg.model.domain*

DAOs

Los DAOs son objetos que encapsulan y abstraen del acceso a los datos a los objetos que implementan la lógica de negocio, que son los que harán uso de estos para las operaciones de escritura, lectura, actualización y borrado de datos, sin conocer detalles de su implementación interna.

Los DAOs de los que consta la aplicación y sus especificaciones están representados en el siguiente diagrama de clases:

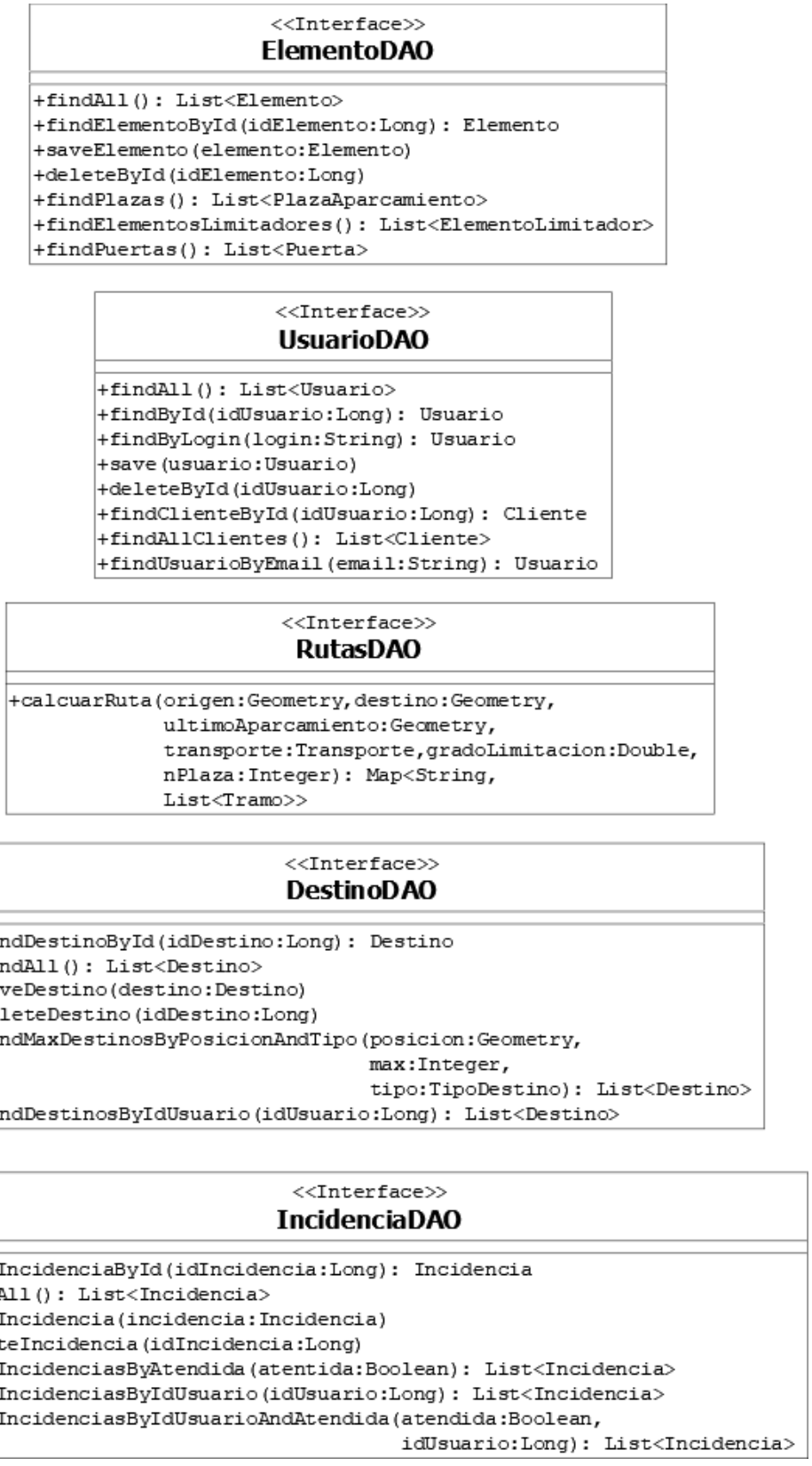


Figura 5.2: Diagrama de clases de los DAOs

Los ficheros fuente de los DAOs de la aplicación se encontrarán en el paquete *es.udc.lbd.asi.servidortfg.model.repository*

Servicios

Los servicios son las clases que implementan la lógica de negocio de la aplicación, para cumplir con los requisitos del proyecto, descritos en la Sección 4.1. Se valen de los DAOs para el acceso a datos y sus métodos son llamados desde los controladores.

Las interfaces de los servicios de la aplicación se representan en el diagrama de clases de la figura 5.3

Adicionalmente, en la tabla 5.1 figuran los métodos de cada servicio y los casos de uso descritos en la Sección 4.1.3 que implementa cada uno de ellos. En esta tabla se puede observar que existen métodos de los servicios que no implementan ningún caso de uso (anotados con *N/A* en la columna casos de uso); esto se debe a que, aunque no sean necesarios actualmente por los requisitos del sistema, se decidió crearlos para dejar los servicio más completos para posibles cambios futuros.

Los ficheros fuente con la definición de las interfaces y las implementaciones de los servicios se encuentran en el paquete
es.udc.lbd.asi.servidortfg.model.service

DTOs

Para el intercambio de información entre el servidor y el cliente web que llama al servicio REST se utilizan los DTOs, que son objetos que contienen la información con la que se comunicarán entre sí.

El diagrama de clases de la figura 5.4 describe los DTOs para los elementos de la aplicación. Además existen DTOs para las entidades usuario, destino e incidencia, para la autenticación de los usuarios en la aplicación y para la petición del cálculo de rutas.

El código en el que se implementan los DTOs se encuentra en el paquete
es.udc.lbd.asi.servidortfg.model.service.dto

Controladores

Los controladores serán los componentes encargados del manejo de la comunicación entre el cliente y el servidor web. Gestionaran las peticiones HTTP que lleguen a cada recurso, llamando a los métodos del servicio para realizar las operaciones y utilizando los DTOs anteriormente mencionados para el intercambio de información entre cliente y servidor.

En las tablas 5.2, 5.3, 5.4, 5.6, 5.7, 5.8, 5.9 y 5.10 figuran los métodos que gestionan cada petición HTTP en cada uno de los controladores y sus URLs de acceso.

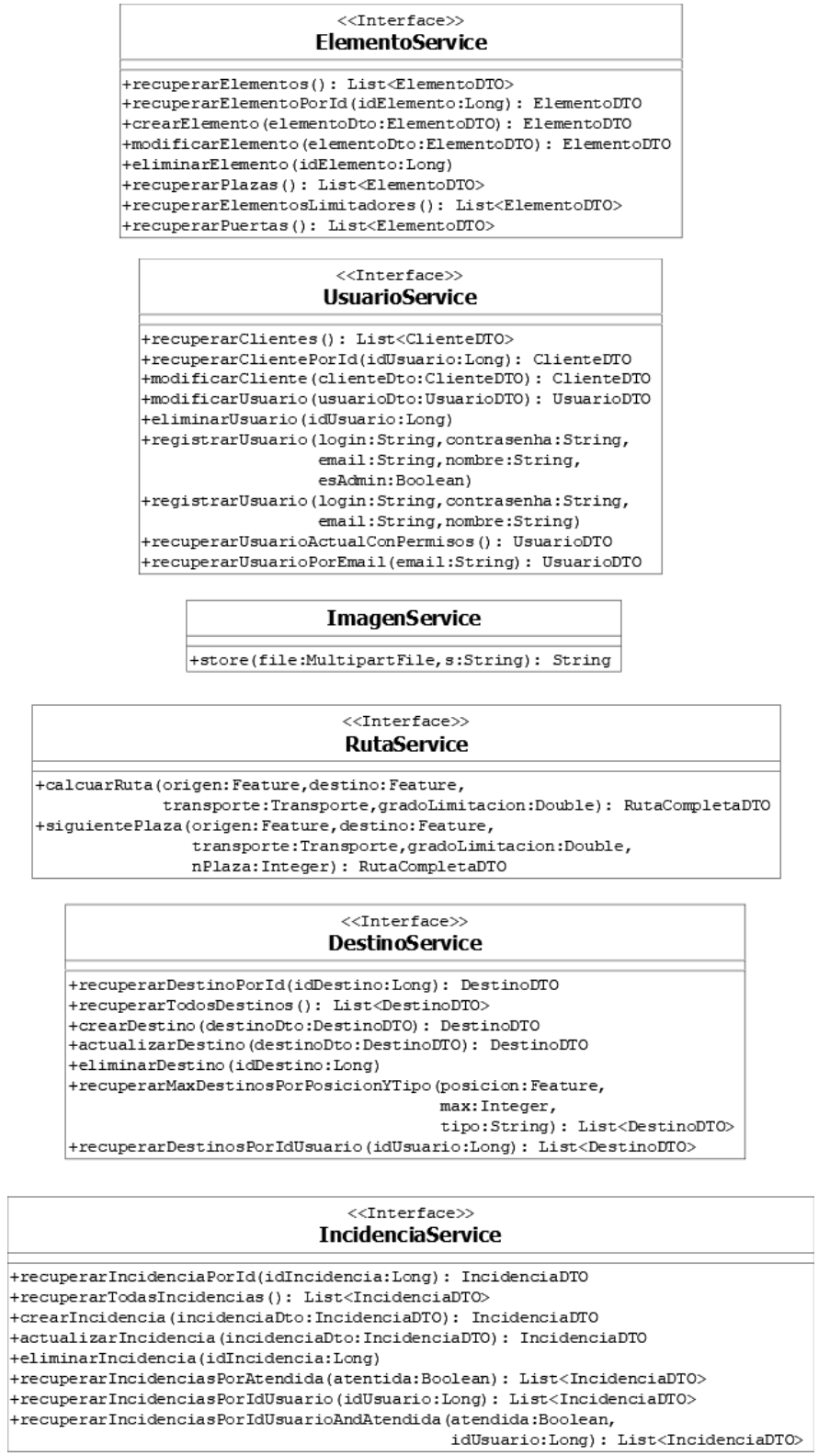


Figura 5.3: Diagrama de clases de los servicios

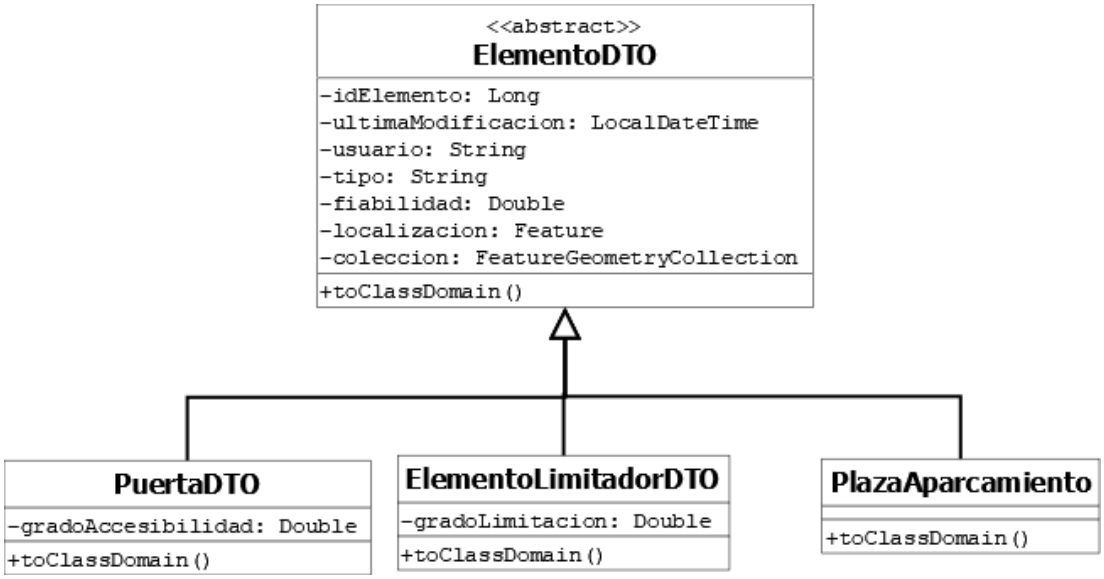


Figura 5.4: Diagrama de clases de los DTOs de los elementos

Los ficheros fuente correpondientes a los controladores del servidor web se encuentran en el paquete *es.udc.lbd.asi.servidortfg.web*.

5.2.2 Cliente web

En esta sección se detallará la estructura de los ficheros fuente, se asocian los componentes que forman la vista a las pantallas de la aplicación diseñadas descritas en la Sección 4.3 y se enumeran las rutas de estos.

Estructura del código

Los componentes Vue con los que se implementa el cliente web se dividen en dos directorios:

- *src/app/entities*: este paquete se divide a su vez varios paquetes (uno por cada entidad de la aplicación) que contienen los componentes relacionados con los datos de las entidades del servidor web.

- *src/app/components*: en este paquete se encuentran tanto los componentes relacionados con pantallas que no manejan datos de las entidades de la aplicación como los que conforman la estructura de la interfaz de usuario.

El diagrama de paquetes de la figura 5.5 representa la distribución de los componentes en los paquetes del cliente web.

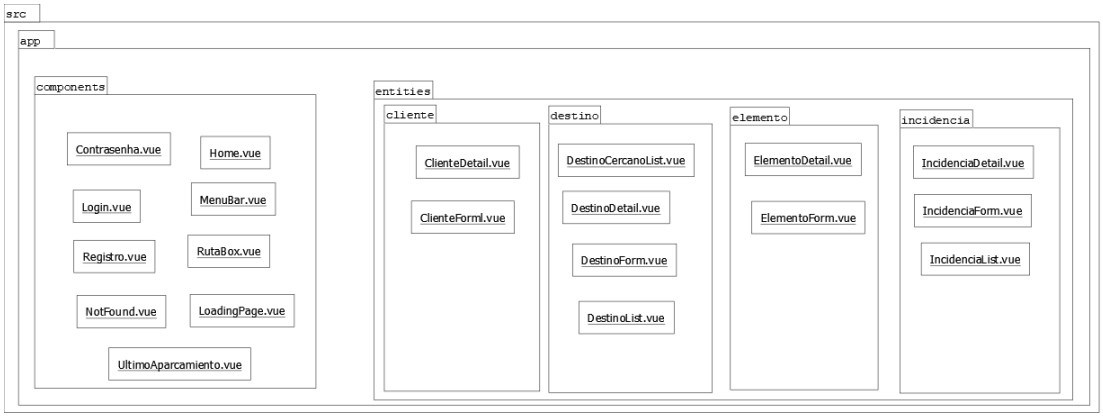


Figura 5.5: Diagrama de paquetes de los componentes Vue del cliente web

Componentes

En la tabla 5.11 se asocia cada componente a cada pantalla diseñada durante el análisis del proyecto, con su URL de acceso. Los componentes son elementos HTML reutilizables (un mismo componente se puede utilizar en distintos contextos modificando ligeramente su comportamiento) con los que se desarrolla la interfaz de la aplicación para ahorrar costes, al reutilizar código repetitivo en la aplicación. Estos componentes se pueden reutilizar a su vez dentro de otros componentes y se pueden comunicar entre sí mediante eventos y propiedades.

Servicio	Método	Casos de uso
ElementoService	recuperarElementos	Visualizar mapa
	recuperarElementoPorId	Detalle elemento
	crearElemento	Crear elemento
	modificarElemento	Modificar elemento
	eliminarElemento	Eliminar elemento
	recuperarPlazas	N/A
	recuperarPuertas	N/A
	recuperarElementosLimitadores	N/A
UsuarioService	recuperarElementos	Visualizar mapa
	recuperarClientes	N/A
	recuperarClientePorId	Ver perfil de usuario
	modificarCliente	Actualizar perfil de usuario
		Registrar trabajo
		Registrar casa
		Marcar último aparcamiento
	eliminarUsuario	N/A
	registrarUsuario	Registrarse
	recuperarUsuarioActualConPermisos	Autenticarse
	recuperarUsuarioPorEmail	Recuperar contraseña
ImagenService	store	Actualizar perfil de usuario
RutaService	calcularRuta	Marcar origen
		Marcar destino
		Calcular ruta punto a punto
		Visualizar mapa
		Empezar navegación
	siguientePlaza	Siguiente plaza
DestinoService	recuperarDestinoPorId	Destino detalle
	recuperarTodosDestinos	Listar destinos Visualizar mapa
	crearDestino	Crear destino
	modificarDestino	Actualizar destino
		Asociar puerta
		Desasociar puerta
	eliminarDestino	Eliminar destino
	recuperarMaxDestinosPorPosicionYTipo	Ver destinos cercanos
	recuperarDestinosPorIdUsuario	Ver destinos usuario
IncidenciaService	recuperarIncidenciaPorId	Incidencia detalle
	recuperarTodasIncidencias	Listar incidencias
	crearIncidencia	Crear Incidencia
	actualizarIncidencia	Actualizar incidencia
	eliminarIncidencia	Eliminar incidencia
	recuperarIncidenciasPorAtendida	Visualizar mapa
	recuperarIncidenciasPorIdUsuarioYAtendida	Ver incidencias usuario
	recuperarIncidenciasPorIdUsuario	N/A

Cuadro 5.1: Casos de uso implementados por los métodos del servicio.

Petición HTTP	Método Java	URL
POST	authenticate	/api/cuentas/authenticate
GET	getAccount	/api/cuentas
POST	registerAccount	/api/cuentas/register
POST	requestPassword	/api/cuentas/password

Cuadro 5.2: Métodos del recurso AccountResource.

Petición HTTP	Método Java	URL
GET	findAllDestinosCliente	/api/cuentas/clientes/idUsuario/destinos

Cuadro 5.3: Métodos del recurso ClienteDestinoResource.

Petición HTTP	Método Java	URL
GET	findAllIncidenciasCliente	/api/cuentas/clientes/idUsuario/incidencias?atendida

Cuadro 5.4: Métodos del recurso ClienteIncidenciasResource.

Petición HTTP	Método Java	URL
GET	findAll	/api/clientes
GET	findClienteById	/api/clientes/idUsuario
PUT	actualizarCliente	/api/clientes/idUsuario

Cuadro 5.5: Métodos del recurso ClienteResource.

Petición HTTP	Método Java	URL
GET	findAll	/api/destinos
GET	findDestinoById	/api/destinos/idDestino
POST	crearDestino	/api/destinos
PUT	actualizarDestino	/api/destinos/idDestino
DELETE	crearDestino	/api/destinos/idDestino
POST	findMaxDestinosByPosicion	api/destinos/posicionytipo?max

Cuadro 5.6: Métodos del recurso DestinoResource.

Petición HTTP	Método Java	URL
GET	findAll	/api/elementos?clase
GET	findElementoById	/api/elementos/idElemento
POST	crearElemento	/api/elementos
PUT	actualizarElemento	/api/elementos/idElemento
DELETE	deleteElementoById	/api/elementos/idElemento

Cuadro 5.7: Métodos del recurso ElementoResource.

Petición HTTP	Método Java	URL
GET	getImageAsResource	/api/imagenes/picRoute
POST	handleFileUpload	/api/imagenes?file

Cuadro 5.8: Métodos del recurso ImagenResource.

Petición HTTP	Método Java	URL
GET	findAll	/api/incidencias?atendida
GET	findIncidenciaById	/api/incidencias/idIncidencia
POST	crearIncidencia	/api/incidencias
PUT	actualizarIncidencia	/api/incidencias/idIncidencia
DELETE	deleteIncidencia	/api/incidencias/idIncidencia

Cuadro 5.9: Métodos del recurso IncidenciaResource.

Petición HTTP	Método Java	URL
POST	calcularRuta	/api/rutas
POST	siguientePlaza	/api/rutas/siguiente/nPlaza

Cuadro 5.10: Métodos del recurso RutaResource.

Componentes	Pantallas	URLs
Contrasenha	Recuperar contraseña	/contrasenha
Home	Pantalla principal	/
LoadingPage	N/A	N/A
Login	Acceso	/login
MenuBar	N/A	N/A
NotFound	N/A	cualquier URL no existente
Registro	Registro	/registro
UltimoAparcamiento	Marcar aparcamiento	/aparcamiento/:coordenadas
ClienteDetail ClienteForm	Perfil de usuario Editar perfil	/clientes/:id /clientes/:id/editar
DestinoCercanoList	Listado de destinos	/destinosproximos
DestinoDetail	Destino Detalle	/destinos/:id
DestinoForm	Formulario destino	/destino/crear/:x/:y
		/destino/:id/editar
DestinoList	Listar destinos	/destinos
ElementoDetail	Elemento Detalle	/elementos/:id
ElementoForm	Formulario Elemento	/elementos/crear/:x/:y
		/elementos/:id/editar
IncidenciaDetail	Incidencia Detalle	/incidencia/:id
IncidenciaForm	Formulario Incidencia	/incidencia/crear/:x/:y
		/incidencias/:id/editar
IncidenciaList	Listar incidencias	/incidencias

Cuadro 5.11: Relación de componentes con sus rutas de acceso y pantallas que implementan.

Capítulo 6

Implementación y pruebas

En este capítulo se explica la lógica de los algoritmos más complejos que utiliza la aplicación. Además, se describen detalles de la implementación no mencionados en capítulos anteriores, al surgir estos en la fase de implementación de las iteraciones del proyecto, por dificultades que surgieron debido a las libreriás utilizadas.

6.1 Algoritmo de cálculo de rutas

De la fase de implementación de la aplicación, el algoritmo para el cálculo de rutas adaptado a personas con movilidad reducida es la parte más complicada, por lo cual se explicará la lógica de este dividido en tres partes: en primer lugar el funcionamiento del algoritmo para un peatón, a continuación se detalla la lógica que añaden los desplazamientos en vehículo y por último la que añade la búsqueda de siguientes plazas.

6.1.1 Algoritmo de cálculo de rutas: peatón

El algoritmo de cálculo de rutas presenta su funcionamiento más sencillo en el caso de que la petición sea hecha como peatón. En este caso, se crea una tabla temporal, con un nombre generado mediante una secuencia numérica para evitar problemas de concurrencia, en la que se introducen todos los tramos (de la red peatonal) que intersecan con en el rectángulo formado por los puntos origen y destino, con unos márgenes de 10 km de cada lado. La figura 6.1 representa visualmente esta lógica. Además se introducen cuatro tramos ficticios formados por arcos: estos arcos irán desde los puntos orígenes y destinos seleccionados por el usuario hasta los nodos final e inicial de cada tramo (paralelos a estos) que contenga dichos puntos (figura 6.2). De esta forma, se consigue que la ruta se ajuste exactamente a la petición del usuario, sin que se vea truncada o extendida.

De estos tramos, se seleccionan aquellos cuya diferencia entre 1 menos el grado de limitación es menor al grado de limitación del peatón, es decir, aquellos tramos por los que el

Figura 6.1: Representación visual del rectángulo generado para seleccionar los tramos

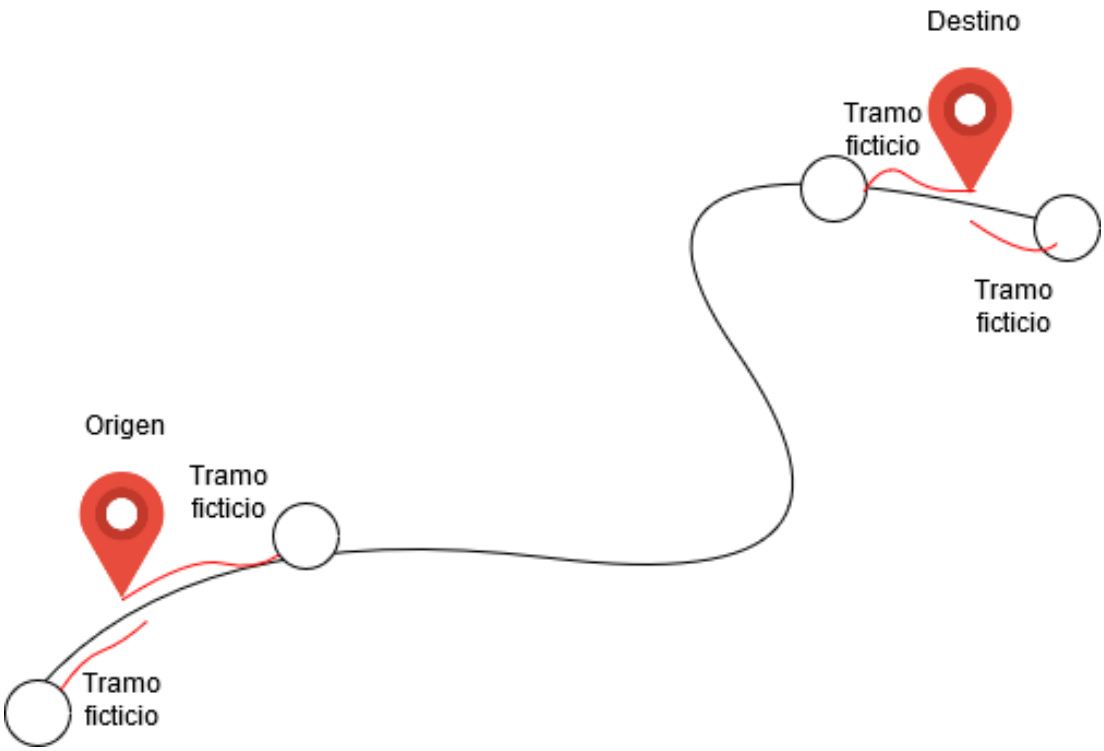


Figura 6.2: Representación visual de los tramos creados en cada petición de cálculo de rutas

peatón puede avanzar.

Con todos los tramos que nos interesan insertados en la tabla temporal, se calcula con el algoritmo de Dijkstra de pgRouting la distancia más corta entre el origen y el destino, asegurándonos de que no se conducirá al usuario por rutas inaccesibles para el, al no estar estos tramos presentes en la tabla temporal.

Por último existe un método para agrupar los tramos por nombres (sólo los tramos consecutivos, ya que es posible ir por una calle y después volver a pasar por ella), sumando sus longitudes para formar así tramos que sean a su vez la unión de varios tramos con el mismo nombre, para proporcionar las indicaciones de la ruta calculada.

6.1.2 Algoritmo de cálculo de rutas: vehículo

En el caso de que el tipo de transporte sea un vehículo, la lógica del algoritmo es más compleja. La ruta a calcular se dividirá entre tres rutas: una ruta será la que lleve al usuario desde su posición actual hasta el punto donde indicó su último aparcamiento; otra ruta llevará al usuario desde el último aparcamiento hasta la plaza de aparcamiento más cercana en un radio de 1 km al punto destino seleccionado; y por último desde esta plaza de aparcamiento hasta la puerta de entrada más cercana (y accesible por el usuario) al destino en un radio de 50

metros. En el caso de no existir último aparcamiento, no existirá la primera ruta; si no existe una plaza cercana se llevará hasta el punto más cercano al destino en la red de carreteras; si no existe ninguna puerta en el destino seleccionado, se llevará hasta el punto más cercano al destino en la red peatonal. La figura 6.3 es una representación de esta lógica.

Al igual que en el caso de la sección 6.1.1, se creará una tabla temporal con los tramos necesarios. La diferencia en este punto entre ambos casos radicará en los tramos seleccionados: se seleccionarán los tramos de la red peatonal al igual en el algoritmo para peatones, tomando como origen el origen seleccionado por el usuario y destino el último aparcamiento del vehículo; los tramos de la red de carreteras que intersequen con el rectángulo formado por los puntos último aparcamiento y plaza de aparcamiento más cercana con un margen de 10 km como en la figura 6.1, sin tener en cuenta el grado de accesibilidad, puesto que se trata de un trayecto en vehículo; y por último los tramos de la red peatonal entre la plaza de aparcamiento y la puerta más cercana al destino, de la misma forma que en el algoritmo de patones de la sección 6.1.1.

Una vez seleccionados estos tramos, se procede con el cálculo del camino más corto accesible para el usuario en cada una de las tres rutas. A continuación se agrupan los tramos en cada una de las tres rutas por separado como en la sección 6.1.1 y se devuelve la ruta global calculada, dividida en los tres trozos mencionados.

6.1.3 Algoritmo de siguiente plaza

Para el caso de uso siguiente plaza, el algoritmo es el mismo que el descrito en la sección 6.1.2 con algunas pequeñas variaciones.

Una de estas variaciones la constituye el parámetro *nPlaza*, que es un contador de las veces que el usuario solicita recalcular la ruta a la siguiente plaza más cercana. Este parámetro se usa para seleccionar a la plaza a la que se dirige la nueva ruta: se almacena en una lista todas las plazas más cercanas al destino en un radio de 1.500 m, y se escoge aquella que esté en la posición del parámetro *nPlaza* mod longitud de la lista, para asegurar que si el número de solicitudes de siguiente plaza sobrepasa al número de plazas, se vuelva a comenzar la lista de plazas, en lugar de lanzar un mensaje de error.

La otra variación es que se suprime la primera ruta de la sección 6.1.2, al asumir que si se solicita recalcular la ruta a otra plaza, es que el usuario ya se encuentra en el vehículo, por lo que sería erróneo conducirlo hasta su último aparcamiento.

6.2 Disparador en las operaciones de escritura de los elementos

Para que la aplicación sea adaptable una vez en producción a los cambios en los elementos, se deben modificar los grados de limitación de los tramos a los que afecten estos cambios. El

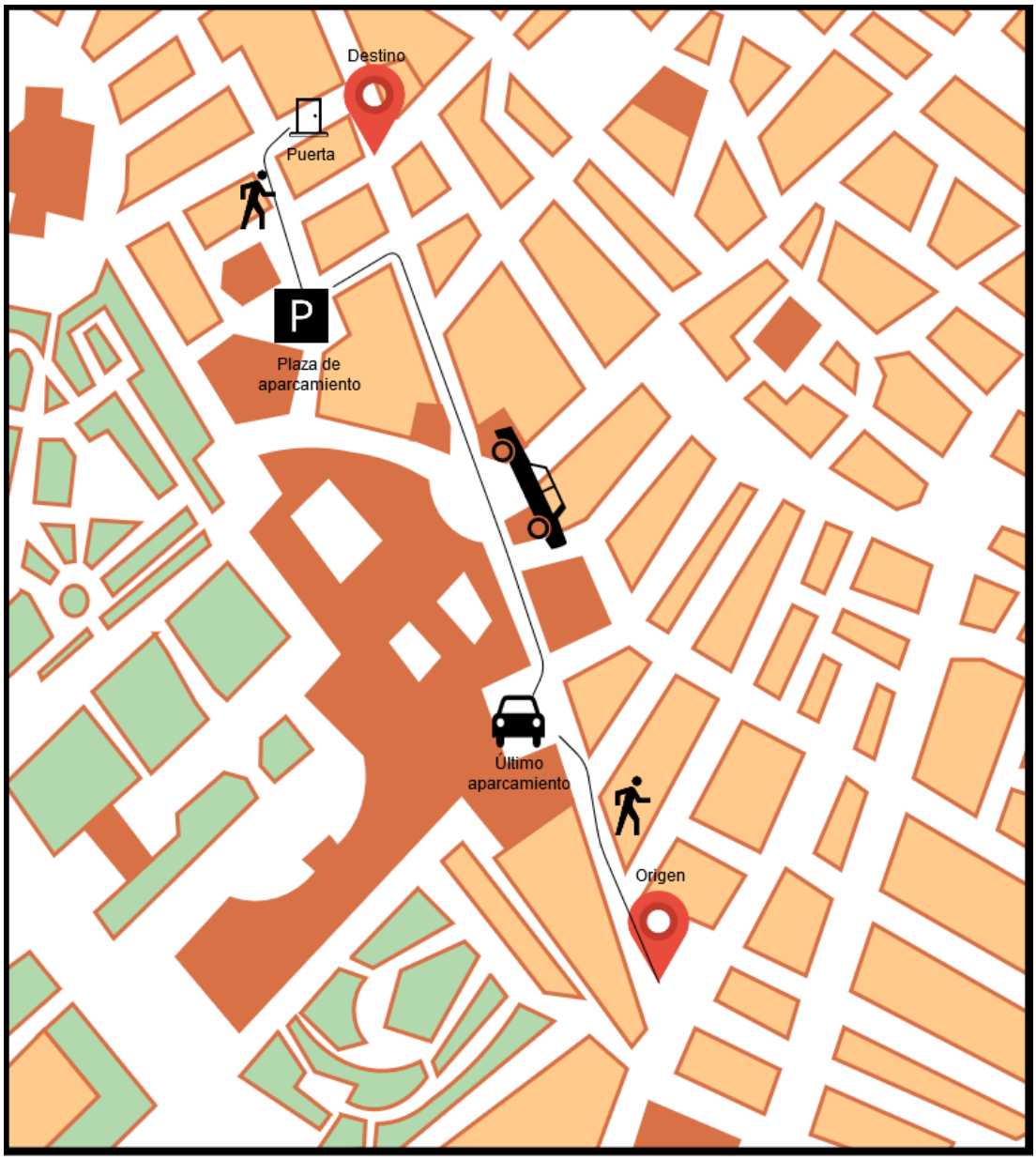


Figura 6.3: Representación visual de las rutas calculadas cuando el transporte es un vehículo

grado de limitación de un tramo viene dado por el mayor grado de limitación de todos los elementos que afecten el tramo.

En este contexto, se diseña un disparador para las operaciones de inserción, actualización y borrado en la tabla de elementos. Este disparador se encarga de actualizar el grado de limitación de los elementos afectados por cada una de estas operaciones.

En primer lugar, el disparador buscará todos los tramos en un radio de 2 metros de la localización del elemento (con esto evitamos que el usuario tenga que emplazar el elemento en el punto exacto del tramo al crear el elemento), que serán los tramos a los que afecta el elemento a insertar/modificar/eliminar.

A continuación, se itera por cada uno de los tramos afectados:

- Si la operación es de inserción o actualización, y el grado de limitación del elemento insertado/actualizado es mayor que el del tramo actual, se modifica el valor del tramo actual al del elemento a insertar/actualizar.
- Si la operación es de actualización y se actualiza el grado de limitación del elemento que marcaba el grado del tramo (esto es, el elemento cuyo grado de limitación es el mayor de entre todos los que afectan al tramo), se actualiza el grado de limitación del tramo al siguiente elemento con un valor más alto en este campo.
- Si se trata de un borrado en la tabla de elementos, y este elemento era el que marcaba el grado de limitación del tramo, se actualiza al grado de limitación del siguiente elemento con un valor más alto en este campo.

Con este procedimiento se asegura que la aplicación sea adaptable a la evolución de la información una vez sea puesta en producción.

6.3 Carga inicial de tramos y elementos

Para la puesta en marcha de la aplicación, se deben insertar los tramos y nodos que forman las dos redes que se utilizan: la de peatones y la de carreteras. Estos tramos se importan de Open Street Maps, con la herramienta OSM2PO.

En primer lugar, se debe configurar la herramienta OSM2PO en el fichero *osm2po.config* para cargar los datos de cada red. Se deberá cambiar la configuración según la red a importar: si es la red de carreteras, se seleccionaran todas las carreteras, calles, autovías y autopistas por las que los vehículos puedan circular; en el caso de la red peatonal, se configurará para importar todas las carreteras, calles y caminos por las que un peatón pueda transitar (todos a la misma velocidad máxima, puesto que se asume que un peatón andará igual de rápido por un camino que por una carretera ou una calle). En la figura 6.4 se muestra la configuración de la herramienta para la carga de la red peatonal.

```
199  #.default.wtr.tag.highway.motorway =       1,  11, 120, foot
200  #.default.wtr.tag.highway.motorway_link =  1,  12, 30,  foot
201  .default.wtr.tag.highway.trunk =          1,  13, 1,   foot
202  .default.wtr.tag.highway.trunk_link =     1,  14, 1,   foot
203  .default.wtr.tag.highway.primary =        1,  15, 1,   foot
204  .default.wtr.tag.highway.primary_link =   1,  16, 1,   foot
205  .default.wtr.tag.highway.secondary =      1,  21, 1,   foot
206  .default.wtr.tag.highway.secondary_link = 1,  22, 1,   foot
207  .default.wtr.tag.highway.tertiary =       1,  31, 1,   foot
208  .default.wtr.tag.highway.tertiary_link =  1,  32, 1,   foot
209  .default.wtr.tag.highway.residential =    1,  41, 1,   foot
210  .default.wtr.tag.highway.road =           1,  42, 1,   foot
211  .default.wtr.tag.highway.unclassified =   1,  43, 1,   foot
212
213  .default.wtr.tag.highway.service =        1,  51, 1,    foot
214  .default.wtr.tag.highway.living_street =  1,  63, 1,    foot
215  .default.wtr.tag.highway.pedestrian =     1,  62, 1,    foot
216  .default.wtr.tag.highway.track =          1,  71, 1,   foot
217  .default.wtr.tag.highway.path =           1,  72, 1,   foot
218  #.default.wtr.tag.highway.cycleway =       1,  81, 1,   foot
219  .default.wtr.tag.highway.footway =        2,  91, 1,    foot
220  .default.wtr.tag.highway.steps =          2,  92, 1,    foot
221  #.default.wtr.tag.route.ferry =            2,   1, 10,  ferry
222  #.default.wtr.tag.railway.rail =           3,   3, 50,  rail
```

Figura 6.4: Trozo del fichero de configuración *osm2po.config* para la carga de la red peatonal

Para la carga desde Open Street Maps, se selecciona el área del que se quieren importar los tramos en la web de OSM y se descarga el fichero xml correspondiente. Despúes, con la herramienta OSM2PO (con la configuración adaptada a la red a cargar), se inicia el proceso de importación en la base de datos de la aplicación. En este caso, los tramos se almacenan en las tablas `ac_2po_4pgr` (red de carreteras) y `rp_2po_4pgr` (red peatonal); análogamente los nodos estarán en las tablas `ac_2po_4pgr_vertices_pgr` y `rp_2po_4pgr_vertices_pgr`.

Tras realizar la carga, se seleccionan los datos a insertar en las tablas de la aplicación Tramo y Nodo, con el atributo red correspondiente según la red a la que pertenezcan.

Por último, durante el paso de los datos de OSM2PO a las tablas de la aplicación, se seleccionan todos aquellos tramos de la red peatonal que sean escaleras (atributo *clazz* = 92) y se insertan como elementos limitadores.

6.4 Detalles de la implementación

En esta subsección se comentarán algunas particularidades de la implementación no contempladas en el análisis ni en el diseño.

6.4.1 Clase GenericDAOHibernate

Esta clase, de la que heredan todos los DAOs de la aplicación, se utiliza para instanciar la clase *SessionFactory* de Hibernate y así poder utilizarla en los DAOs, sin necesidad de declararla en cada uno de ellos.

6.4.2 Conversión DTO a entidad y viceversa

Durante el desarrollo, se observó la necesidad de crear una clase que funcionase como conversor entre los elementos y los DTOs. Su funcionamiento consiste en determinar que instancia de elemento se trata (puerta, plaza o elemento limitador) y devolver el DTO que corresponde según la clase de la entidad.

Para la conversión del JSON enviado por el cliente al DTO correspondiente a la entidad se hace uso de los paquetes *com.fasterxml.jackson.annotation.JsonSubTypes* y *com.fasterxml.jackson.annotation.JsonTypeInfo*, que a través del valor del atributo clase instancian automáticamente el DTO que corresponda.

Cada uno de estos DTOs constan de un método que devuelve la entidad con la información sacada del DTO, de forma que el código es mantenible y apto para cambios futuros, ya que la posible creación de una clase nueva de elemento no alteraría el código existente.

6.4.3 Feature y conversores entre Feature y Geometry

Para la comunicación de geometrías entre cliente y servidor, se utiliza el formato GeoJSON. Al tratarse las geometrías de tipos complejos, es necesario definir una clase de utilidad que represente a una Feature, del formato GeoJSON, en Java.

Para convertir las Features recibidas al tipo *Geometry* de JTS, se ha creado una clase de utilidad que implementa toda la lógica relativa a esta casuística. En ella se cogen las coordenadas de cada Feature y el tipo de esta para devolver la geometría correspondiente.

En el caso inverso, convertir las geometrías Java en *Features* del formato GeoJSON, se crea otra clase de utilidad. Esta clase se encarga de leer las coordenadas y el tipo de cada *Geometry* JTS, y crea el objeto Feature correspondiente.

Estas clases también soportan las conversiones de las *geometry collections*. La clase *FeatureGeometryCollection* representa una geometría de este tipo en formato GeoJSON, y en el DTO de elementos existe un atributo coleccion de este tipo para dar soporte a que un elemento pueda tener como localización una *geometry collection*.

6.4.4 Conversor grados de limitación/accesibilidad

En los DTOs que contienen atributos de grado de limitación o accesibilidad, estos son cadenas de texto (así vienen de la vista al seleccionarlos el usuario).

En las entidades, estos atributos se almacenan como valores numéricos (*double*), por lo que se crea una clase de utilidad, con el mapeo de las cadenas llegadas desde la vista a los valores numéricos que se persistirán de cada entidad. Este mismo proceso se realiza en sentido inverso, para enviar al cliente los grados en cadena de texto.

6.5 Pruebas

Para comprobar el correcto funcionamiento de las funcionalidades implementadas en la aplicación según los requisitos del proyecto, se realizan tres pruebas diferentes sobre ella: pruebas de los servicios, pruebas de los controladores y pruebas de la interfaz de usuario

6.5.1 Pruebas unitarias de los servicios

En el nivel más bajo de las pruebas realizadas se comprueba que los métodos de cada uno de los servicios funcionan correctamente. Para esto se realizaron pruebas unitarias [36], de la mano de la herramienta JUnit.

Estas pruebas consisten en probar de formas aislada un componente software, para verificar que esté se comporta de forma correcta ante distintas entradas, sin tener en cuenta interacciones con otros componentes de la aplicación.

La implementación de estas pruebas se realizó mediante tests JUnit sobre cada uno de los servicios del *backend*. En ellas se comprobó que los métodos devolvían la salida esperada ante determinadas entradas, que lanzaban las excepciones necesarias ante entradas erróneas y que dichos métodos sólo podían ser llamados, en los casos que fuese necesario, por usuarios autenticados y con un determinado rol.

En la figura 6.5 se ve la ejecución de todos los tests unitarios de todos los servicios, con resultado exitoso.

6.5.2 Pruebas sobre los controladores

En un nivel más alto en las pruebas detalladas en la sección 6.5.1, se encuentran las pruebas sobre los controladores, las cuales se realizaron con la herramienta Postman.

Para el desarrollo de dichas pruebas, se lanzaron peticiones al servicio REST, mediante la herramienta mencionada, simulando las llamadas que hace el cliente web al servidor. Se realiza una llamada por cada caso de prueba, de forma que se pueda verificar el correcto funcinamiento del servidor antes distintas entradas.

En la figura 6.6, se ve un ejemplo de llamada al servicio REST desde la herramienta Postman. Como se puede apreciar, se elige la url de acceso, el método HTTP y el cuerpo de la petición (cuando se necesario) en formato JSON para realizar las peticiones.

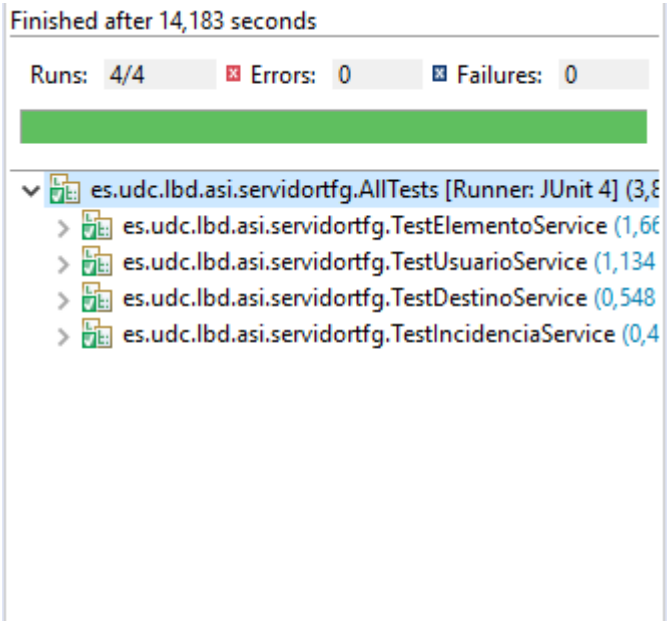


Figura 6.5: Ejecución de los tests JUnit de los servicios

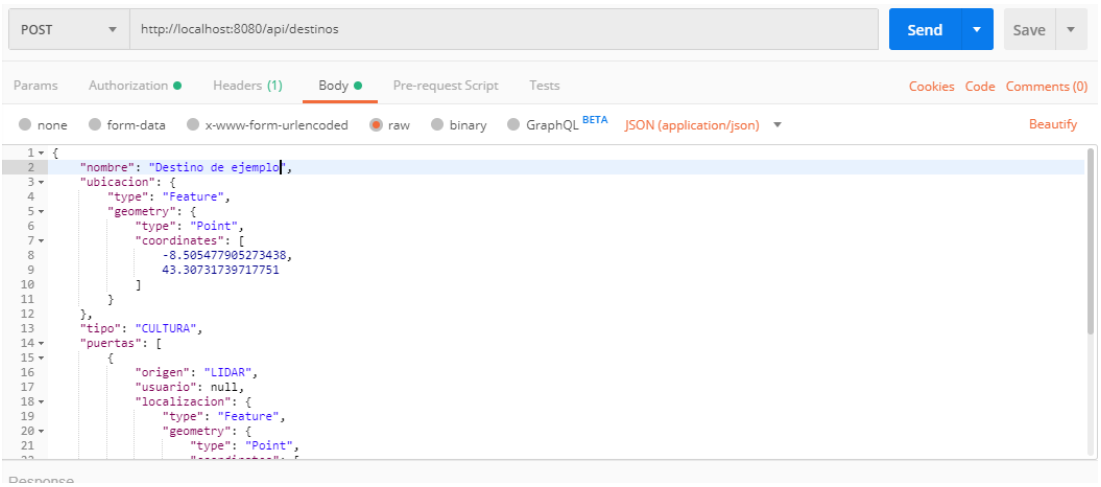


Figura 6.6: Ejemplo de llamada con Postman

6.5.3 Pruebas de la interfaz de usuario

En el nivel más alto de las pruebas realizadas, se encuentran las de la interfaz de usuario. Se tratan de pruebas funcionales [37], comprobando que el software final desempeña el funcionamiento esperado en los casos de uso que implementa este.

La ejecución de estas pruebas se basa en las ejecuciones de los casos de uso por parte del analista/programador de este proyecto, comprobando visualmente que este devuelve la salida esperada ante las distintas entradas de cada caso de uso.

Capítulo 7

Solución desarrollada

En este capítulo se realizará una guía por las funcionalidades principales del producto final desarrollado a lo largo del proyecto, entre las que se incluyen las funcionalidades de gestión de usuario, las operaciones de gestión de información de las distintas entidades de la aplicación y el cálculo de rutas

7.1 Gestión de usuarios

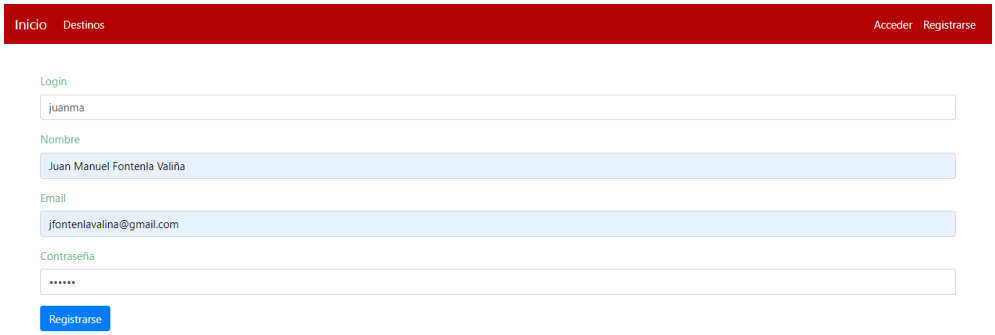
Relacionada con la gestión de usuarios, se encuentran en primer lugar el registro y la autenticación de un usuario en la aplicación. Estas funcionalidades estarán accesibles desde la barra de navegación de la aplicación. Como se ve en la figura 7.1b, existe un enlace que dirige a la pantalla de recuperación de contraseña, en la que se deberá introducir el email al que llegará la nueva clave de acceso, como se ve en la figura 7.1c

Una vez autenticado como usuario, se puede navegar hasta el perfil de este y modificarlo. Se podrá acceder a esta funcionalidad desde el enlace de la barra de navegación con el *login* del usuario. En la figura 7.2 se ve el perfil de un usuario y el formulario para la modificación de sus datos, que cuenta con mapas para marcar las ubicaciones de la casa y el trabajo.

7.2 Operaciones con elementos

Las operaciones que pueden realizar los usuarios no administradores con los elementos son únicamente las de lectura de estos. Para consultar la información de un elemento del mapa, basta con clicar en el para que aparezca una barra lateral con los datos relativos al mismo, como se ve en la figura 7.3a

Sin embargo, un administrador puede, desde la vista detalle de un elemento, acceder a la edición del mismo o eliminarlo, a través de los botones que aparecen en la figura 7.3b. En el caso de pulsar el botón eliminar, saldrá un mensaje de advertencia como el de la figura 7.4a

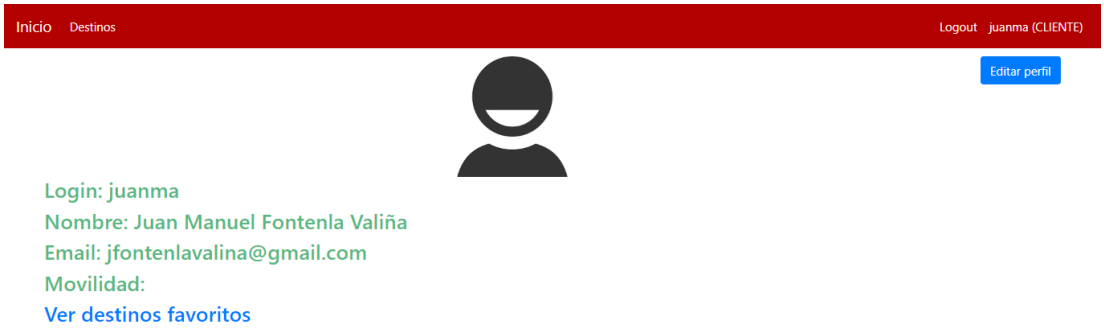


(a) Registro

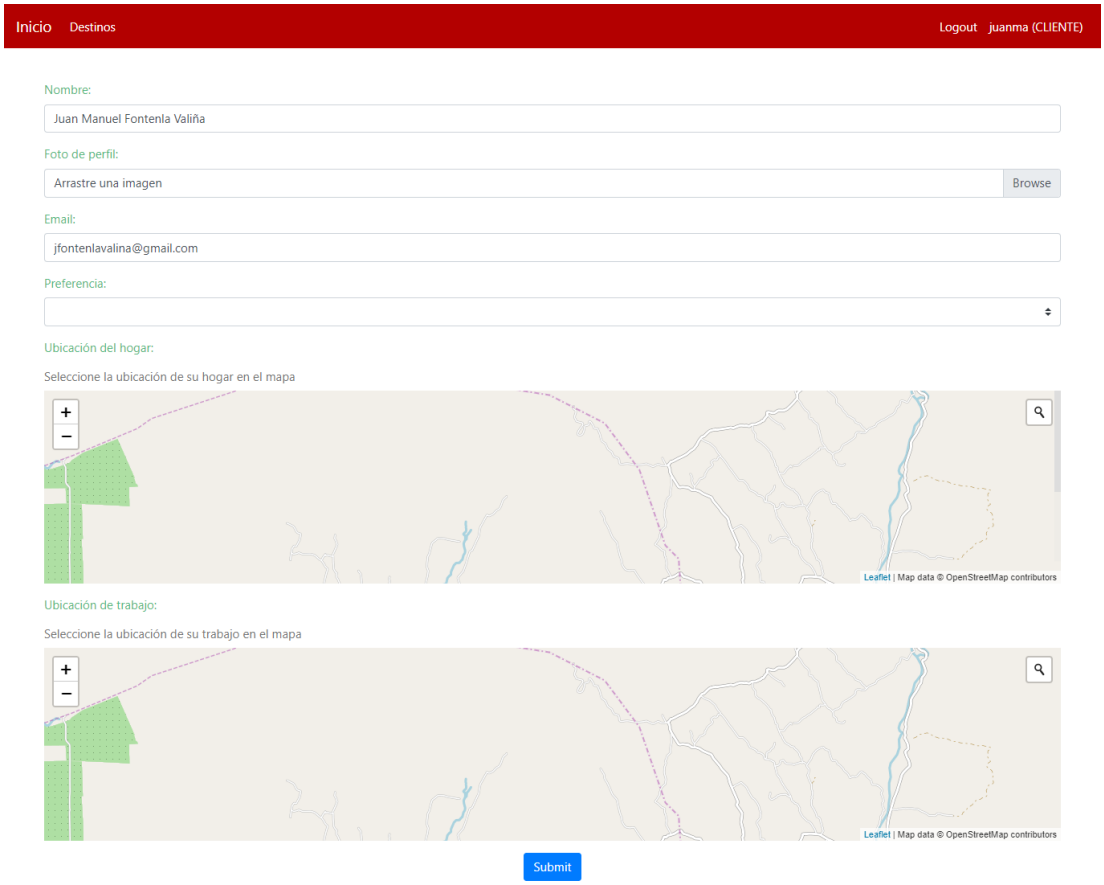
(b) Autenticación

(c) Recuperación de contraseña

Figura 7.1: Gestión de usuarios de un usuario no autenticado

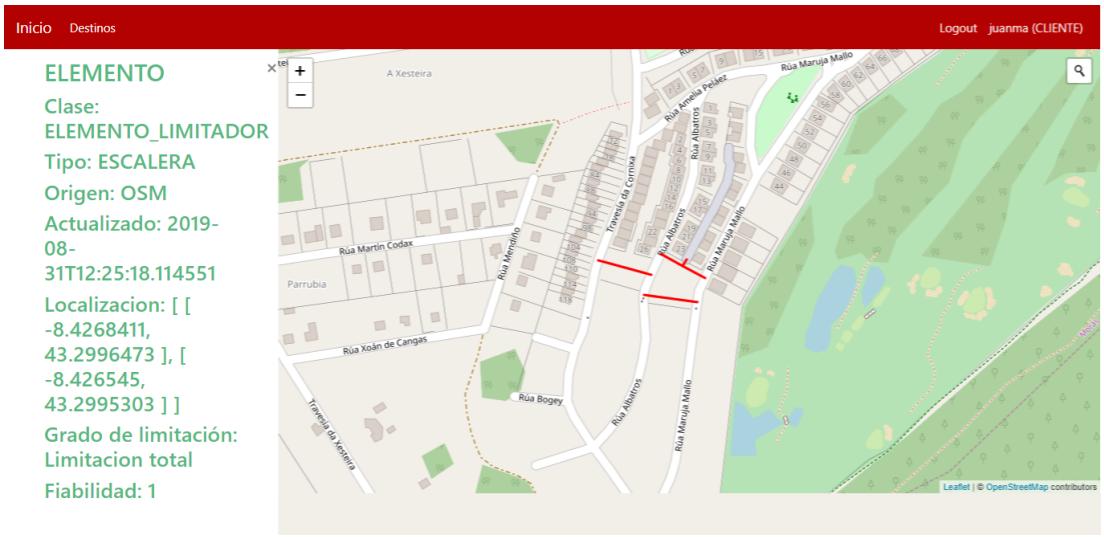


(a) Perfil de usuario

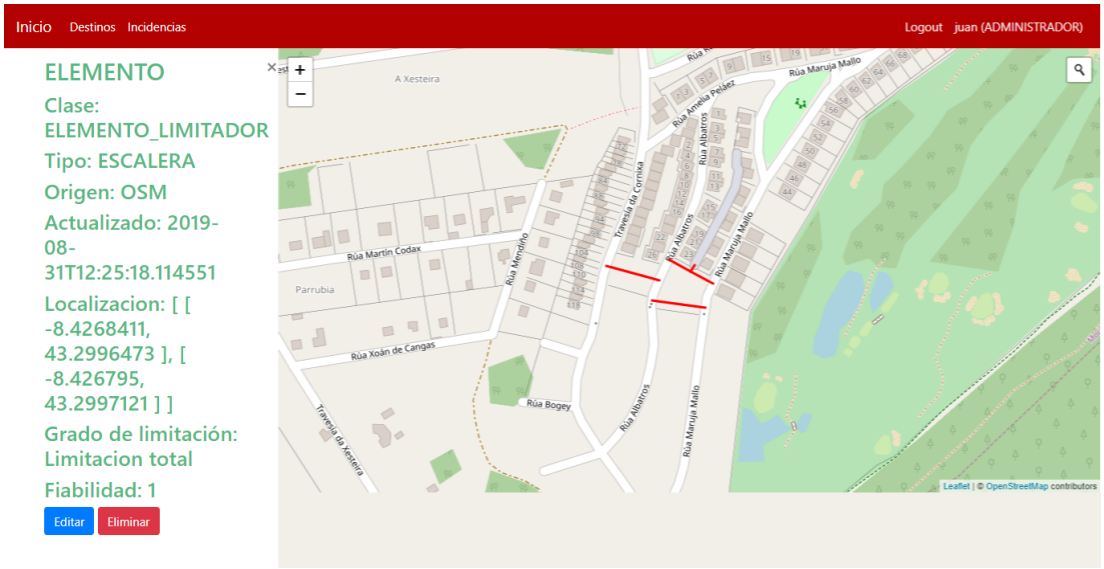


(b) Formulario de modificación del perfil de usuario

Figura 7.2: Gestión del perfil del usuario



(a) Detalle de un elemento: usuario no administrador



(b) Detalle de un elemento: administrador

Figura 7.3: Detalle de un elemento

para confirmar la acción. La edición del elemento llevará al formulario de la figura 7.4b de este para modificar los datos, que variará según el tipo de elemento seleccionado.

Además, un administrador tiene la posibilidad de crear un elemento desde el menú de opciones del mapa (figura 7.4c). Al seleccionar la opción crear elemento, aparecerá en la barra lateral el formulario de alta del elemento (el mismo que el ya mencionado formulario de modificación de la figura 7.4b), con el elemento ubicado en el punto en el que el usuario selecciona la opción.

7.3 Operaciones con destinos

En lo referente a los destinos, las funcionalidades de gestión de información de los mismos (crear, actualizar, ver detalle y eliminar) son similares a las ya detalladas en la sección 7.2, como se ve en las figuras 7.5.

Además, en el detalle del destino se puede centrar la visión del mapa a este y marcarlo como punto origen o punto destino de una posible ruta a calcular. Si el usuario tiene el rol de cliente, podrá marcar el destino como favorito (figura 7.5a).

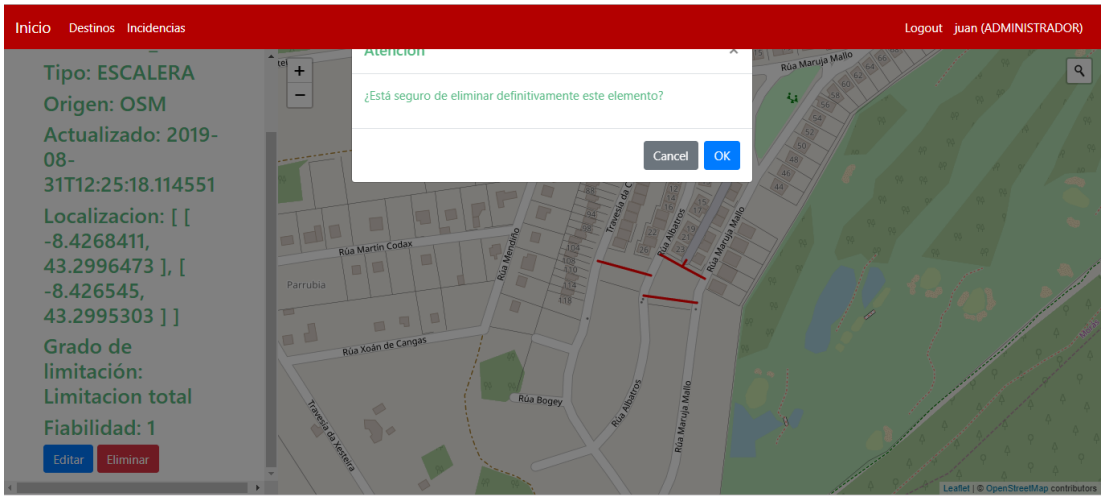
A diferencia de los elementos, existe un listado en el que se pueden ver todos los destinos registrados en la aplicación (figura 7.6a). Además, en la figura 7.6b se observa otro listado, esta vez de los destinos seleccionados como favoritos por parte del usuario, el cual es accesible desde el enlace *Ver destinos favoritos* de la figura 7.2. Desde estos listados se podrá acceder al detalle de cada uno de los destinos del mismo.

Por último, también se puede acceder, desde cualquiera de las listas de destinos, a la pantalla de destinos cercanos. Como se ve en la figura 7.7, en esta pantalla se verán los destinos más cercanos a la posición actual del usuario (5 máximo) de cada tipo, desde el que se podrá acceder al detalle de cada destino.

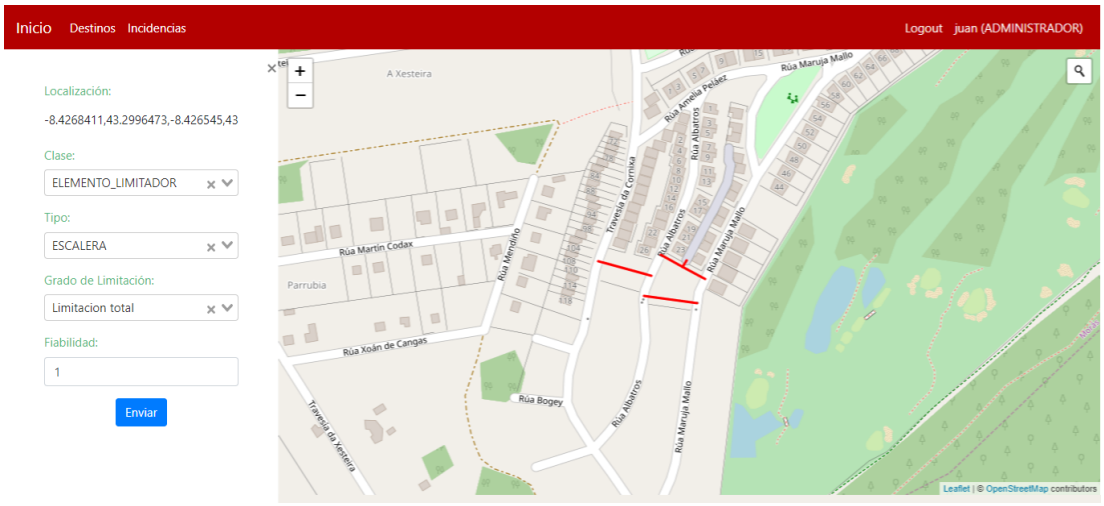
7.4 Operaciones con incidencias

Para el tratamiento de incidencias, la funcionalidad que ofrece la aplicación es equivalente a la vista en las secciones 7.2 y 7.3 para las operaciones de creación, modificación, lectura y borrado (figuras 7.9). La particularidad que presentan estas, es que solo pueden ser creadas por el usuario, que lo hará a través del menú de opciones del mapa (figura 7.8). Además, como en el caso de los destinos, se puede consultar el listado de todas las incidencias que han sido dadas de alta en la aplicación (sólo disponible para usuarios administradores, figura 7.10).

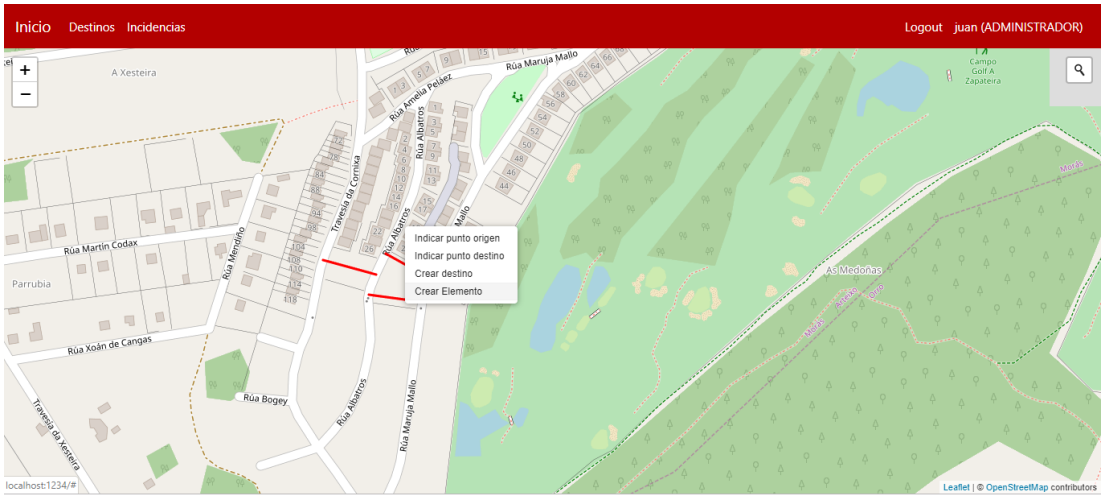
En el mapa principal de la aplicación, los usuarios no registrados no podrán ver ninguna de las incidencias; los usuarios con rol de cliente verán en el mapa todas sus incidencias que no han sido atendidas por algún administrador; y por último los usuarios con rol de administrador



(a) Mensaje de advertencia antes de borrar un elemento

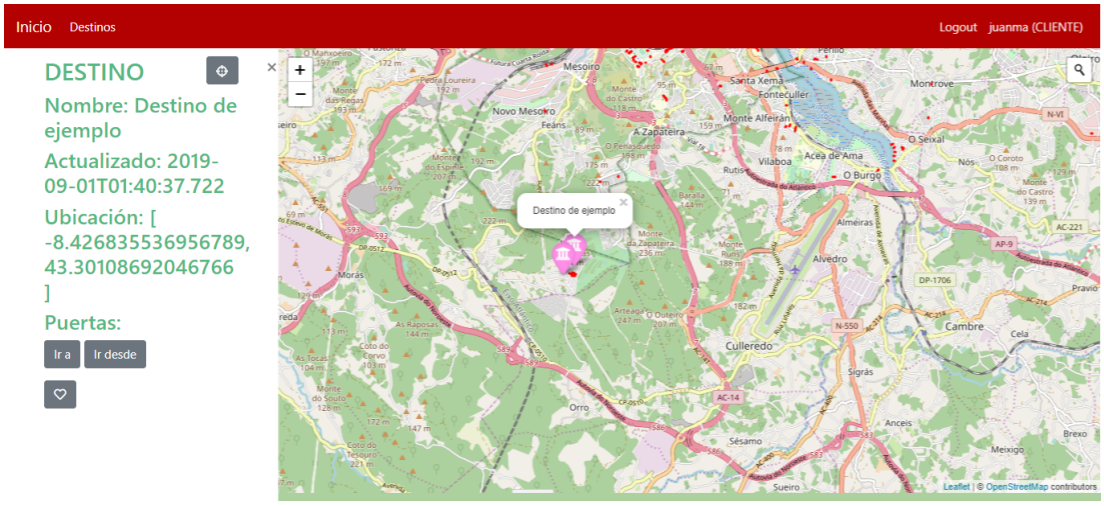


(b) Formulario de un elemento

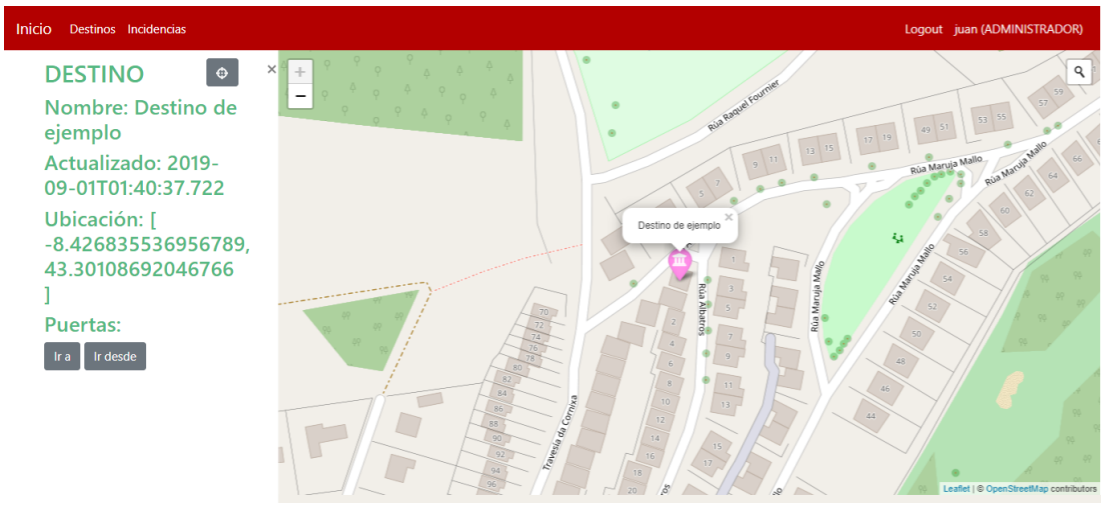


(c) Opción crear elemento

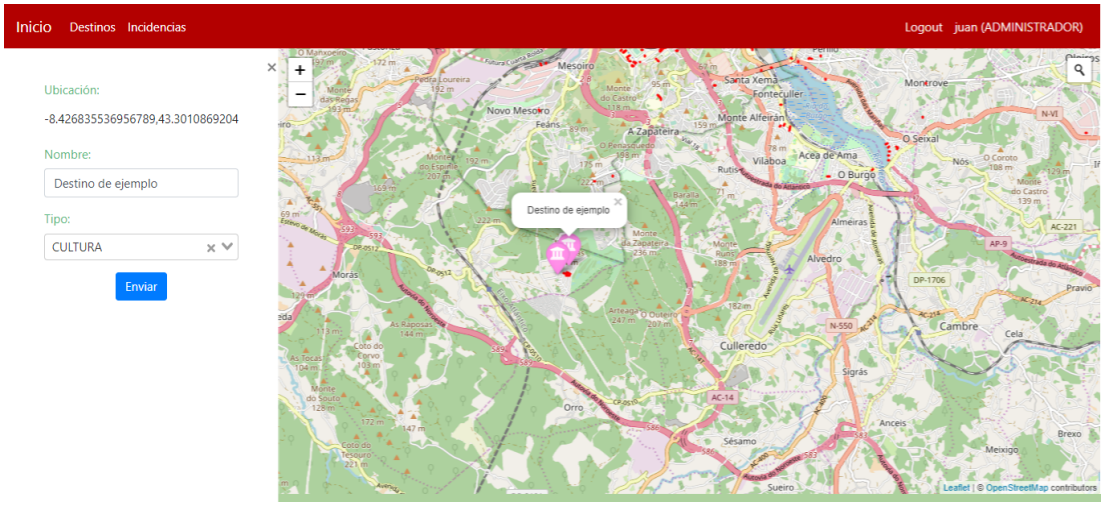
Figura 7.4: Operaciones de escritura de elementos



(a) Detalle de un destino: cliente



(b) Detalle de un destino: administrador



(c) Formulario de un destino

Figura 7.5: Lectura y escritura de destinos

(a) Lista de los destinos de la aplicación

(b) Lista de destinos favoritos de un usuario

Figura 7.6: Listados de destinos

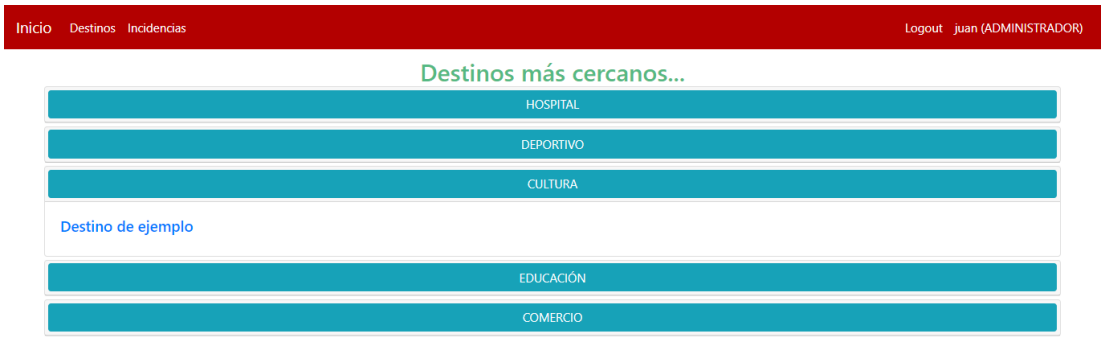


Figura 7.7: Destinos cercanos al usuario

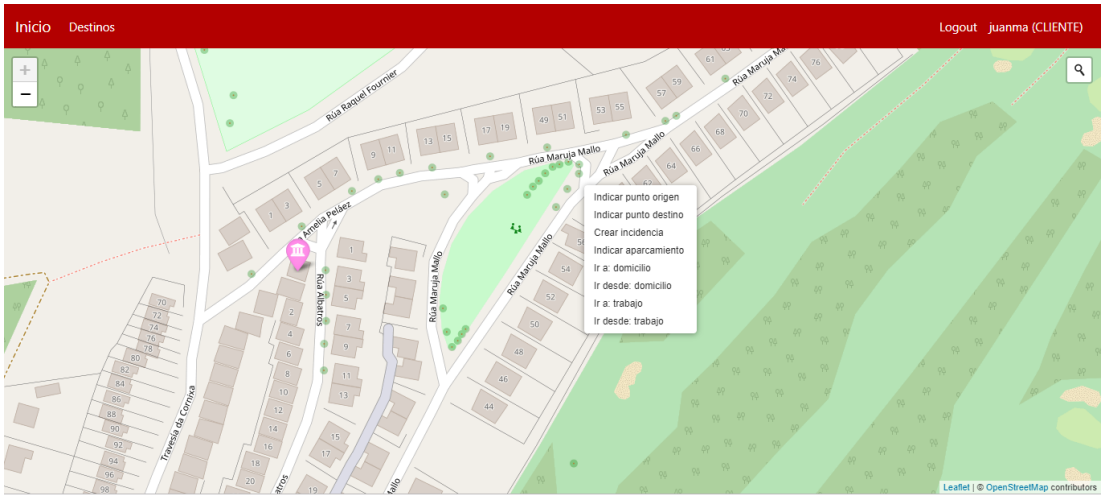
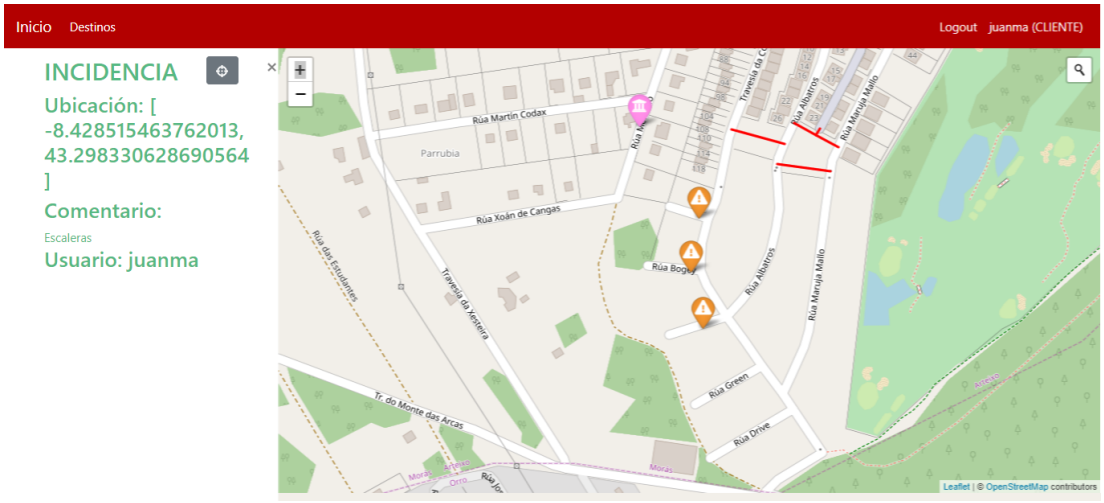
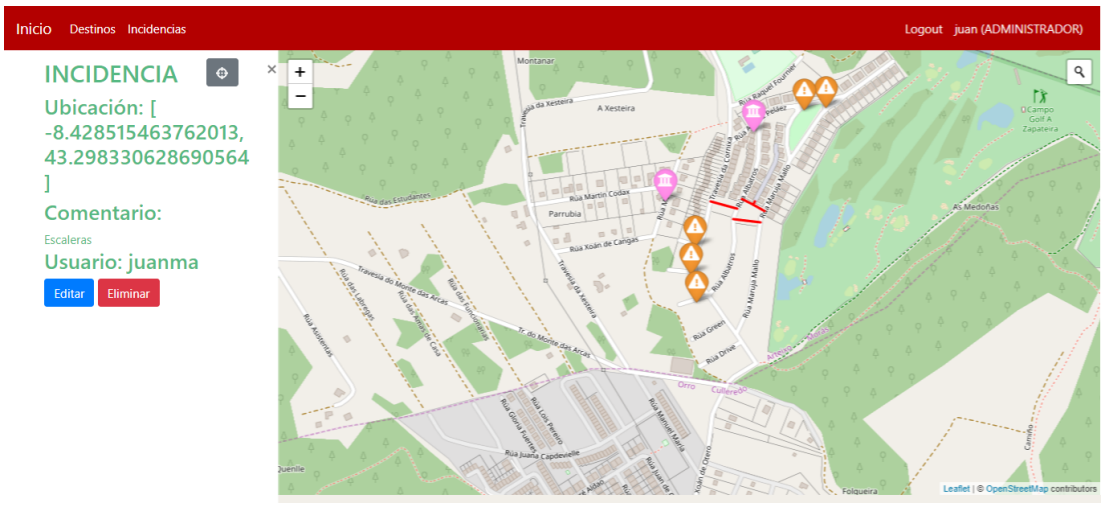


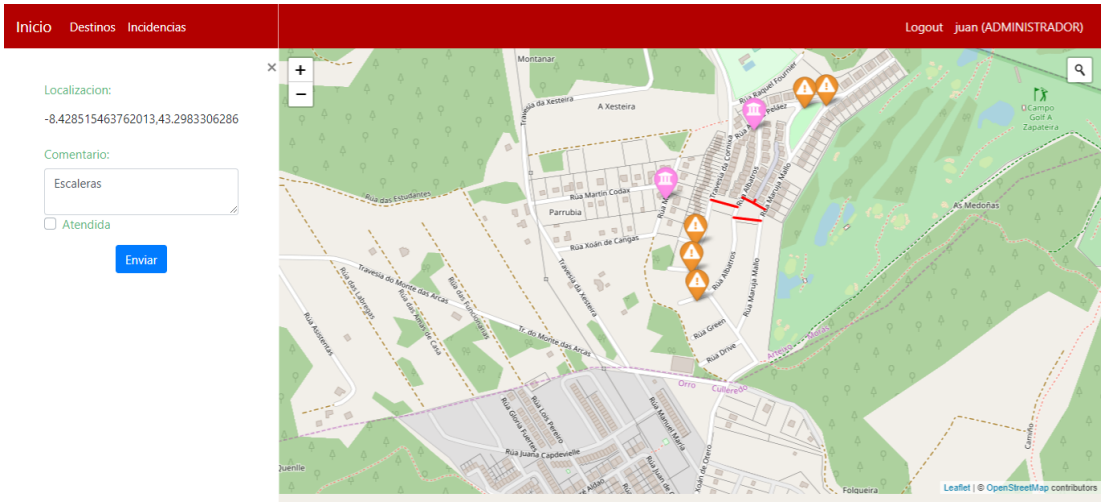
Figura 7.8: Menú de opciones de un usuario cliente



(a) Detalle de una incidencia: cliente



(b) Detalle de una incidencia: administrador



(c) Formulario de alta/modificación de una incidencia

Figura 7.9: Escritura y lectura de incidencias

Figura 7.10: Lista de incidencias de la aplicación

verán en el mapa todas las incidencias dadas de alta por los clientes y que todavía no han sido atendidas por ningún administrador.

7.5 Cálculo de rutas y navegación

En lo referente al apartado de cálculo de rutas se puede acceder a la pantalla para enviar la petición de varios modos:

- Desde el menú de opciones del mapa principal de la aplicación (figura 7.8):
  - En la opción *Indicar punto origen*, que pondrá el punto seleccionado como origen de la ruta. Esta opción está disponible para todos los usuarios.
  - En la opción *Indicar punto destino*, que pondrá el punto seleccionado como destino de la ruta. Esta opción está disponible para todos los usuarios.
  - En la opción *Ir desde: domicilio*, que pondrá el punto que marcó el usuario en algún momento previo como su domicilio, como origen de la ruta. Esta opción está disponible únicamente para usuarios clientes.
  - En la opción *Ir a: domicilio*, que pondrá el punto que marcó el usuario en algún momento previo como su domicilio, como destino de la ruta. Esta opción está disponible únicamente para usuarios clientes.
  - En la opción *Ir desde: trabajo*, que pondrá el punto que marcó el usuario en algún momento previo como su ubicación de trabajo, como origen de la ruta. Esta opción está disponible únicamente para usuarios clientes.

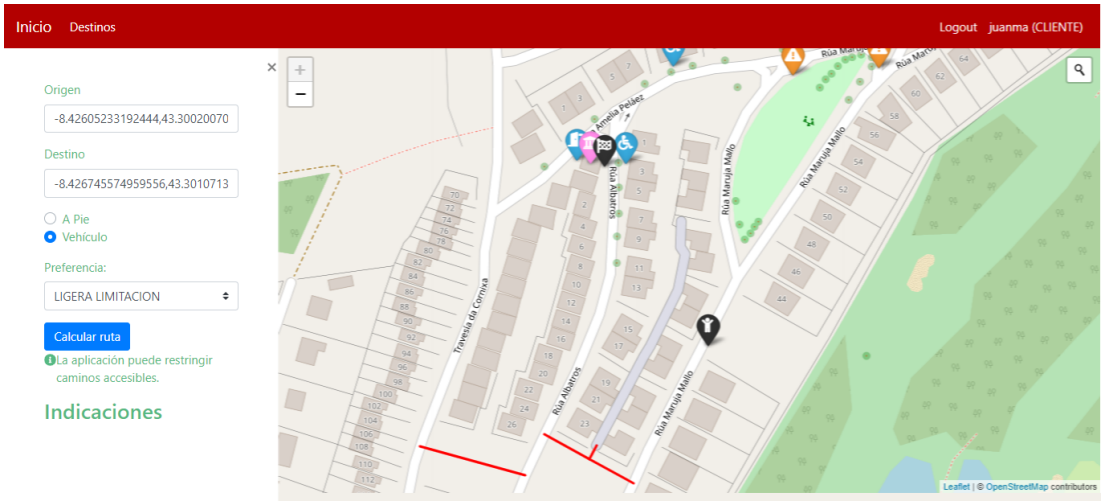


Figura 7.11: Formulario para el cálculo de rutas

  - En la opción *Ir a: trabajo*, que pondrá el punto que marcó el usuario en algún momento previo como su ubicación de trabajo, como destino de la ruta. Esta opción está disponible únicamente para usuarios clientes.

- Desde el detalle de un destino de la aplicación (figura 7.5b):

  - Pulsando el botón *Ir desde*, que pondrá la ubicación del destino como origen de la ruta.
  - Pulsando el botón *Ir a*, que pondrá la ubicación del destino como destino de la ruta.

Una vez realizada alguna de las acciones mencionadas, se desplegará la barra lateral en la pantalla principal con el formulario del cálculo de rutas (figura 7.11). Por defecto, el grado de limitación de este formulario será *SIN LIMITACION* o el grado que haya indicado previamente el usuario (cliente) en su perfil. Se deberá indicar además el tipo de transporte con el que se hará la ruta: vehículo o peatón. Al pulsar el botón *Calcular ruta*, se procederá con el cálculo de la misma en el servidor.

Al calcularse la ruta, aparecen las indicaciones de esta y se pinta en el mapa. Si la ruta era en vehículo, esta estaría troceada en tres partes, como se describió en la sección 6.1.2, diferenciando los trozos en colores diferentes y con un marcador informando del cambio de trozo (figura 7.12). Además aparecerá un botón para iniciar la navegación por la ruta.

Al iniciar la navegación, el mapa se centrará en la posición actual del usuario, y podrán aparecer hasta tres botones en la parte superior (figura 7.13):

- Marcar aparcamiento: si el usuario que realiza la solicitud de cálculo de la ruta tiene

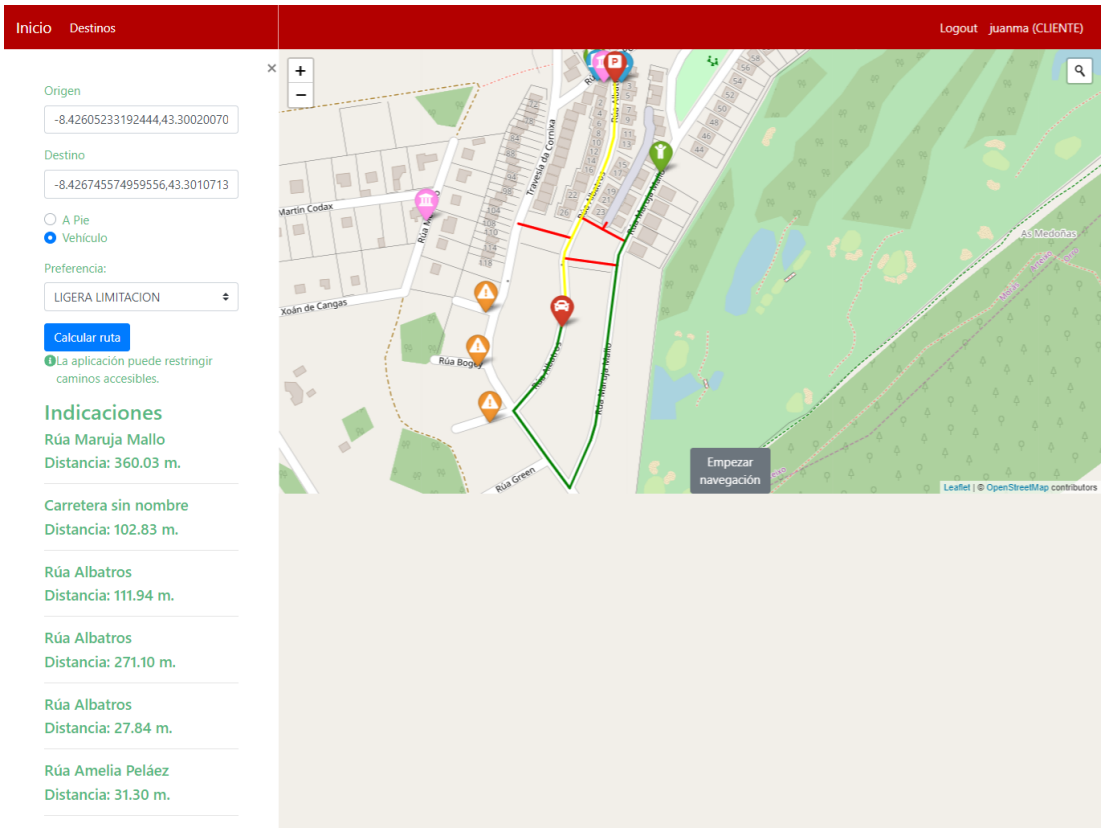


Figura 7.12: Ruta calculada

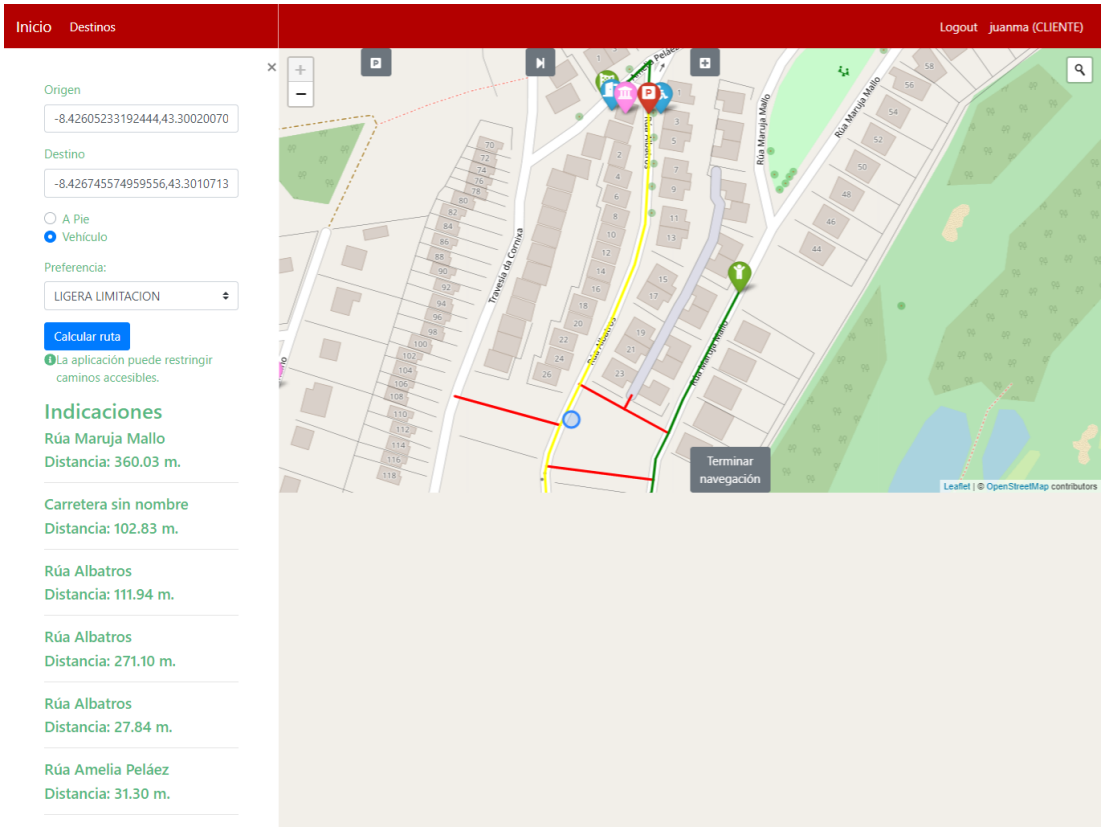


Figura 7.13: Navegación a través de la ruta calculada

el rol de cliente, y esta tiene como tipo de transporte vehículo, podrá marcar el último aparcamiento, que será su posición actual.

- Siguiente plaza: si la ruta a calcular tiene como tipo de transporte vehículo, se mostrará este botón que permite recalcular la ruta a la siguiente plaza de aparcamiento más cercana.

- Crear incidencia: si el usuario que realiza la solicitud de cálculo de la ruta tiene el rol de cliente, se mostrará este botón, que llevará al formulario de creación de una incidencia, que tendrá como ubicación la posición actual del usuario (figura 7.9c).

Al terminar la navegación, la pantalla vuelve a su estado original.

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo se hace una valoración final sobre el proyecto, describiendo el grado de cumplimiento de los objetivos planteados al inicio en la sección 1.2, la utilidad de la solución desarrollada y las lecciones aprendidas en el desarrollo del proyecto. Por último se enumeran una serie de amplicaciones, fuera de los objetivos del proyecto, que pueden ser interesantes de incorporar en un futuro.

8.1 Conclusiones sobre el trabajo final

Al término del trabajo realizado hasta este momento en este proyecto, se concluye que:

- Los objetivos planteados al inicio de este, descritos en la sección 1.2, se cumplen al implementar con éxito toda la funcionalidad detallada en la sección 4.1, con la que se cubren dichos objetivos.

- El producto desarrollado en este proyecto sirve para ofrecer un sistema de información geográfica orientado a personas con movilidad reducida. Podrán consultar las rutas entre dos puntos, adaptada a sus características personales, navegar por la misma que permite ver en tiempo real el trayecto a realizar y aportar información al mismo sobre la accesibilidad de calles y destinos, desde cualquier dispositivo con conexión a Internet.

- Tras el desarrollo de proyecto, se aprendió de esta experiencia la importancia de una buena planificación y la dificultad de realizar esta, al no contar con datos previos de otros proyectos y al usar muchas tecnologías previamente desconocidas durante el desarrollo del proyecto.

  Por otro lado, se entiende la relevancia de las fases previas a la implementación del proyecto (análisis y diseño), así como la necesidad de realizar unas pruebas completas. Las fases de análisis y diseño posibilitan desarrollar un producto mucho más completo

y acorde a los objetivos iniciales, al descomponer los requisitos en casos de uso que se traducen en funcionalidades del producto final

Además, se asimiló la importancia de usar una metodología de desarrollo iterativa, al ver los cambios que se produjeron durante el transcurso del proyecto (al tener que replanificar el mismo y por las dificultades e ideas surgidas durante la implementación) desde la fase de análisis.

En cuanto al apartado técnico, se entendió la ventaja de usar una arquitectura en capas, pudiendo realizar cambios complejos en cualquier capa sin tener la necesidad de tocar el trabajo realizado en las capas inferiores; la utilidad de los disparadores en la base de datos para mantener coherencia en estos; las facilidades de usar librerías como Spring o Vue, que abstraen al programador de la implementación de código repetitivo y tedioso (gestión de las conexiones a base de datos, mapeo de las peticiones HTTP, el código JavaScript para implementar las funcionalidades que incorpora Vue, etc.) para focalizarse en el desarrollo lógica que requiere el proyecto.

8.2 Posibles ampliaciones futuras

Aunque han sido implementadas todas las funcionalidades necesarias para cumplir los requisitos del proyecto, se podrían hacer algunas mejoras en la aplicación, como:

- Crear elementos cuya ubicación no sea un punto, sino una línea o un polígono: actualmente, los elementos que solo pueden ser dados de alta desde la aplicación como puntos, por lo que se podría facilitar al usuario crear elementos con otros tipos geométricos. Respecto al cálculo de rutas, la funcionalidad sería la misma, pero sería una representación más visual para el usuario pintar, por ejemplo, una escalera como una línea en vez de como un punto.
- Notificar la atención de una incidencia: se podría implementar un método para enviar un email a un usuario cuando una incidencia que haya sido reportada sea atendida por un administrador.
- Modificar un usuario su propia incidencia: actualmente un usuario que notifica una incidencia no puede modificarla, para evitar marcarla como atendida, aunque sería posible establecer una verificación de que un usuario cliente pueda modificar sólo de sus propias incidencias no atendidas el comentario de estas.
- Aplicación nativa en Android: desarrollar una vista nativa en Android, además del cliente web, para facilitar el acceso a la aplicación a los posibles usuarios de dispositivos Android, sin necesidad de acceder desde el navegador de estos a la misma.

- Posibilidad de filtrado en el mapa: facilitarle al usuario un filtro en el que pueda seleccionar qué marcadores quiere ver en el mapa (elementos, incidencias y/o destinos) y poder elegir ver solo aquellos elementos limitadores que le impedirián la accesibilidad por su movilidad y aquellas puertas que le permiten el acceso por el mismo criterio.

Bibliografía

[1] "Página web de Google Maps," 2019, (Consultado en enero de 2019). [Online]. Available: https://www.google.es/maps/preview/

[2] "Página web de Mapp4All," (Consultado en enero de 2019). [Online]. Available: https://www.mapp4all.com/

[3] "Página web de Disbled Park," (Consultado en enero de 2019). [Online]. Available: https://www.disabledpark.com/

[4] "Accessibility Plus," (Consultado en enero de 2019). [Online]. Available: https://www.fundacionvodafone.es/apps-accesibles/accessibility

[5] "Página de PostgreSQL," (Consultado en febrero de 2019). [Online]. Available: https://www.postgresql.org/

[6] "Página de PostGIS," (Consultado en marzo de 2019). [Online]. Available: https://postgis.net/

[7] "Página de pgRouting," (Consultado en abril de 2019). [Online]. Available: https://pgrouting.org/

[8] "Página de Hibernate," (Consultado en febrero de 2019). [Online]. Available: https://hibernate.org/

[9] "Página de HibernateSpatial," (Consultado en marzo de 2019). [Online]. Available: http://www.hibernatespatial.org/

[10] "Página de Spring Boot," (Consultado en febrero de 2019). [Online]. Available: https://spring.io/projects/spring-boot

[11] "Artículo sobre GeoJSON," 2016, (Consultado en marzo de 2019). [Online]. Available: https://es.wikipedia.org/wiki/GeoJSON

[12] "Página de JUnit," (Consultado en marzo de 2019). [Online]. Available: https://junit.org/junit5/

[13] "Página de osm2po," (Consultado en abril de 2019). [Online]. Available: https://osm2po.de/

[14] Vladimir Agafonkin, "Página web de Leaflet," 2019, (Consultado en marzo de 2019). [Online]. Available: https://leafletjs.com/

[15] "Página de Vue.js," (Consultado en febrero de 2019). [Online]. Available: https://vuejs.org/

[16] "Página de Node.js," (Consultado en febrero de 2019). [Online]. Available: https://nodejs.org/es/

[17] "Página de BootstrapVue," (Consultado en marzo de 2019). [Online]. Available: https://bootstrap-vue.js.org/

[18] "Repositorio Git del proyecto Axios," (Consultado en marzo de 2019). [Online]. Available: https://github.com/axios/axios

[19] "Desarrollo iterativo incremental," (Consultado en febrero de 2019). [Online]. Available: https://proyectosagiles.org/desarrollo-iterativo-incremental/

[20] "Desarrollo en cascada," (Consultado en agosto de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Desarrollo_en_cascada

[21] Sarah Dámaris Amaro Calderón, Jorge Carlos Valverde Rebaza, "Metodologías Ágiles," 2007, (Consultado en agosto de 2019). [Online]. Available: https://s3.amazonaws.com/academia.edu.documents/43784053/METODOLOGIAS_AGILES.pdf?response-content-disposition=inline%3B%20filename%3DUniversidad_Nacional_de_Trujillo.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWOWYYGZ2Y53UL3A%2F20190905%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20190905T220634Z&X-Amz-Expires=3600&X-Amz-SignedHeaders=host&X-Amz-Signature=7009a44bfa0ba01b169b8847ff95a0fc4d647f888d10581d149eb0e07f65f445

[22] Philippe Kruchten, "Le Rational Unified Process," 1999, (Consultado en agosto de 2019). [Online]. Available: http://profs.etsmtl.ca/claporte/English/Enseignement/CMU_SQA/Lectures/Normes/Rational_Unified_Process_Introduction_Francais_Krutchen.pdf

[23] "Artículo sobre Dia," (Consultado en enero de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Dia_(programa)

[24] "Página de Balsamiq," (Consultado en enero de 2019). [Online]. Available: https://balsamiq.com/

[25] "Página de Maven," (Consultado en febrero de 2019). [Online]. Available: https://maven.apache.org/

[26] "Página de Eclipse," (Consultado en marzo de 2019). [Online]. Available: https://www.eclipse.org/

[27] "Página de npm," (Consultado en marzo de 2019). [Online]. Available: https://www.npmjs.com/

[28] "Página de SublimeText," (Consultado en marzo de 2019). [Online]. Available: https://www.sublimetext.com/

[29] "Página de GitLab del Laboratorio de Bases de Datos," (Consultado en marzo de 2019). [Online]. Available: https://gitlab.lbd.org.es/

[30] "Página de Notepad++," (Consultado en abril de 2019). [Online]. Available: https://notepad-plus-plus.org/

[31] "Página de Postman," (Consultado en marzo de 2019). [Online]. Available: https://www.getpostman.com/

[32] "Actor (UML)," (Consultado en agosto de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Actor_(UML)

[33] "Requisito (sistemas)," (Consultado en agosto de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Requisito_(sistemas)

[34] "Caso de uso," (Consultado en agosto de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Caso_de_uso

[35] "Programación por capas," (Consultado en agosto de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Programación_por_capas

[36] "Prueba unitaria," (Consultado en agosto de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Prueba_unitaria

[37] "Pruebas funcionales," (Consultado en agosto de 2019). [Online]. Available: https://es.wikipedia.org/wiki/Pruebas_funcionales

Apéndices

Apéndice A

Protipos de pantalla

A continuación se añaden los protipos de pantalla realizados en el análisis previo del proyecto, omitidos por razones de espacio en la sección 4.3. Estos prototipos no se corresponden fidedígnamente con el producto final, sino que sirvieron como idea de la interfaz de usuario a implementar para la aplicación.

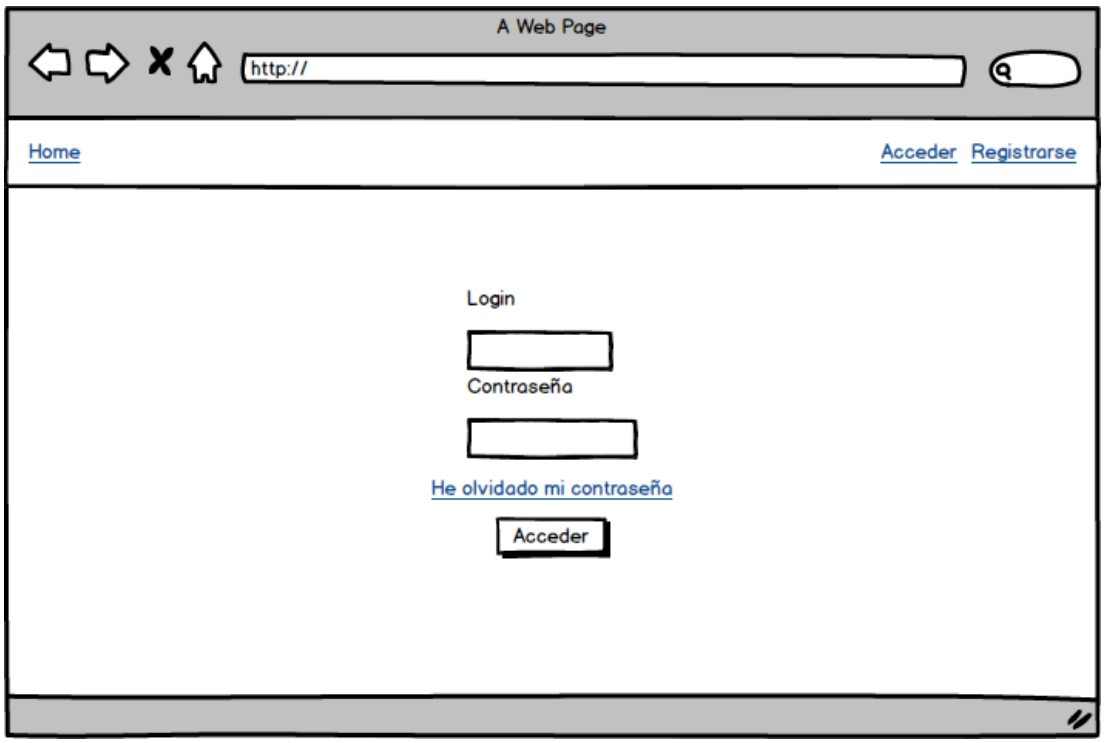


Figura A.1: Acceso

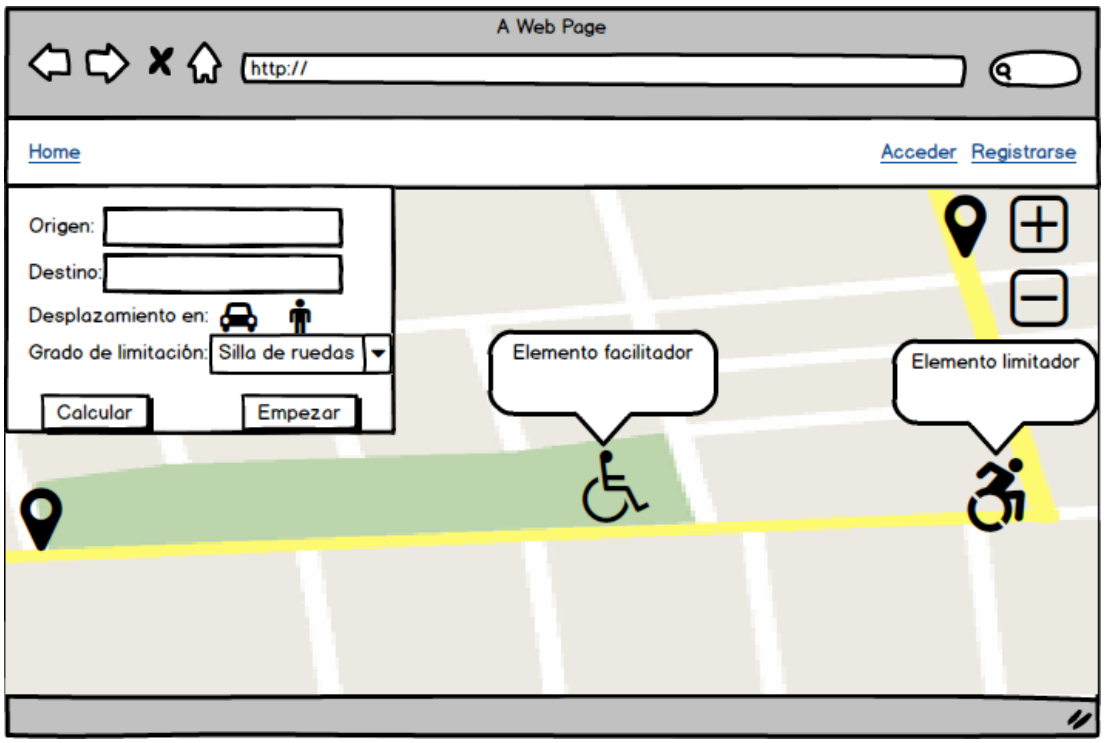


Figura A.2: Cálculo de ruta

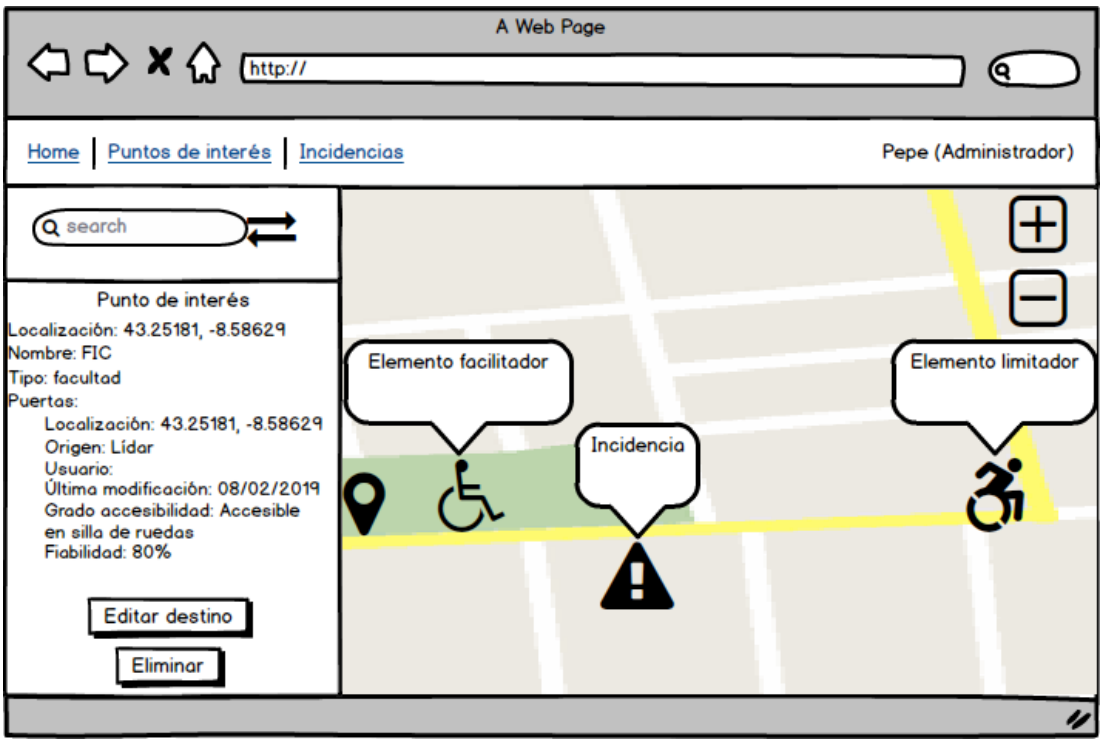


Figura A.3: Destino detalle (administrador)

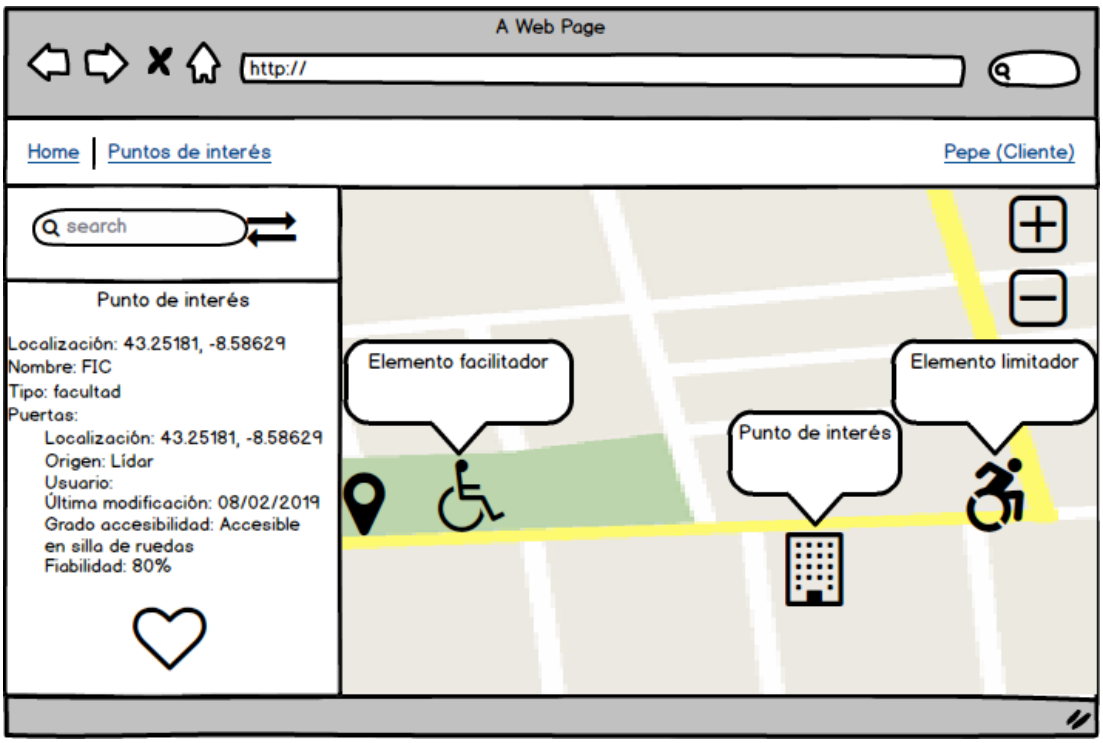


Figura A.4: Destino detalle

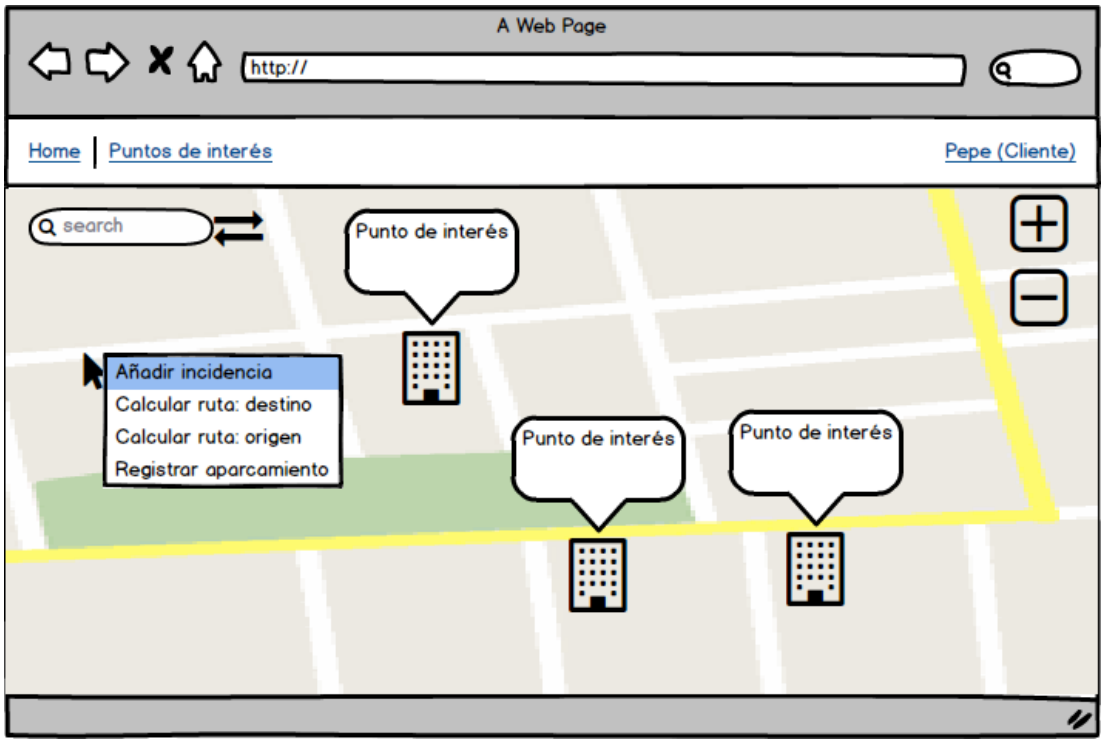


Figura A.5: Destino usuario

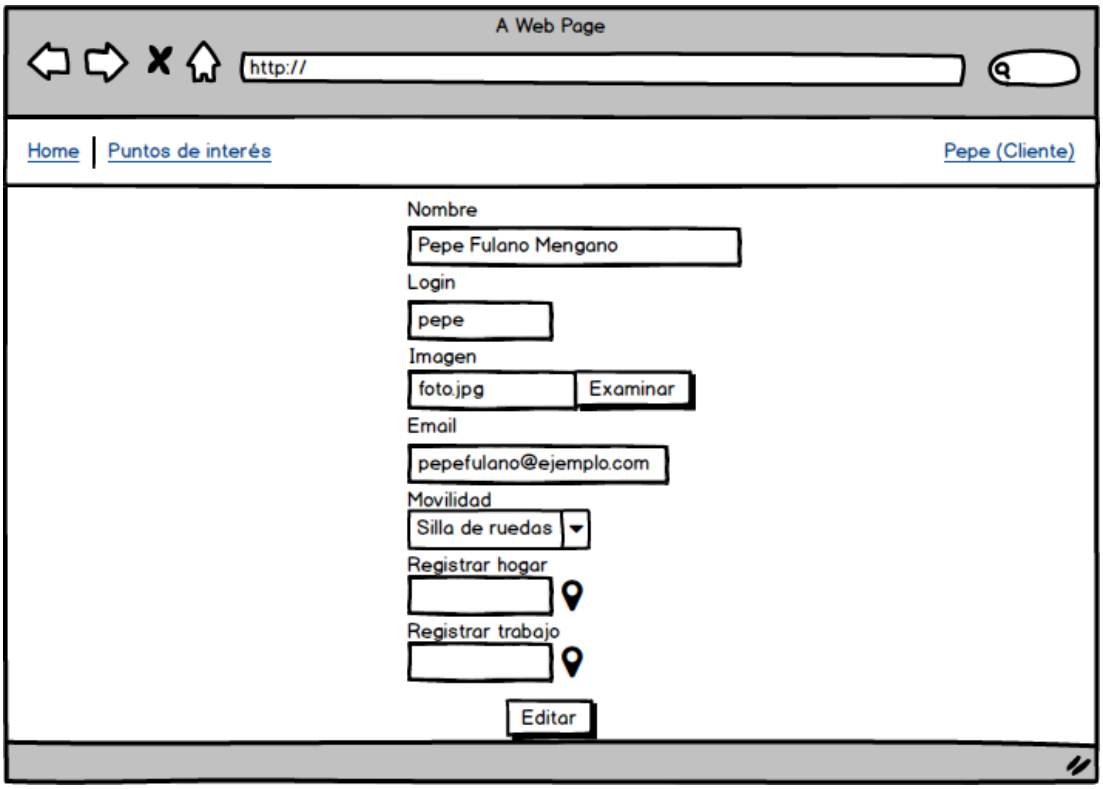


Figura A.6: Editar perfil

Figura A.7: Elemento detalle (administrador)

Figura A.8: Elemento detalle

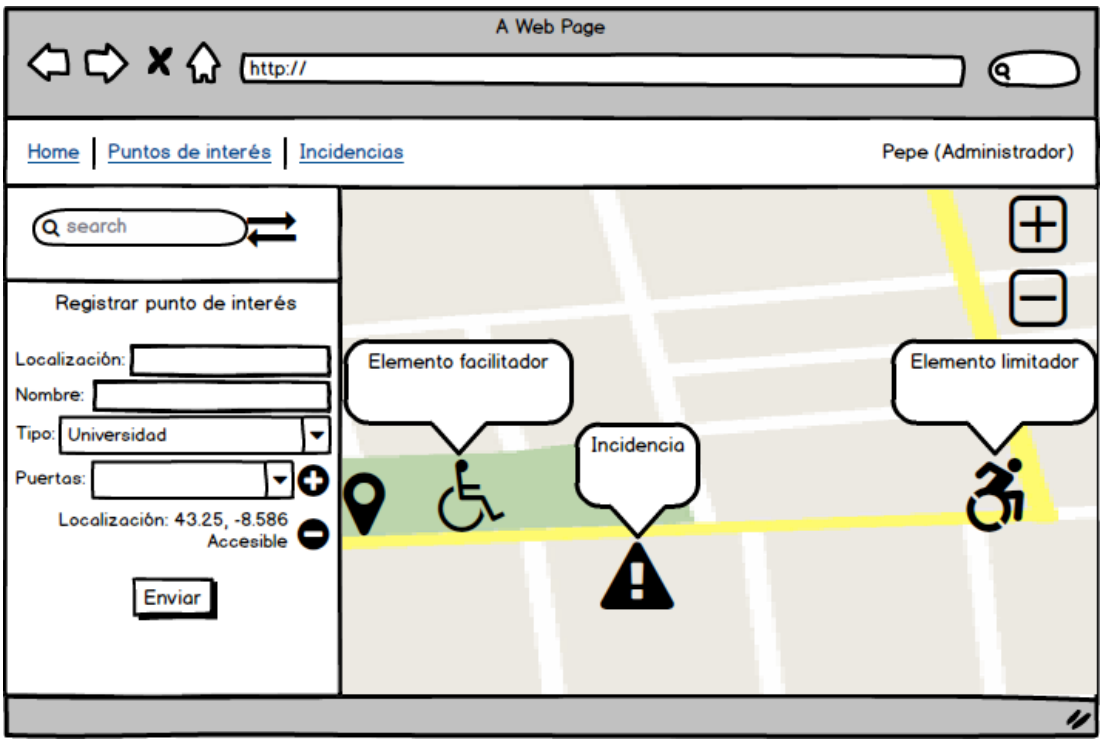


Figura A.9: Formulario destino

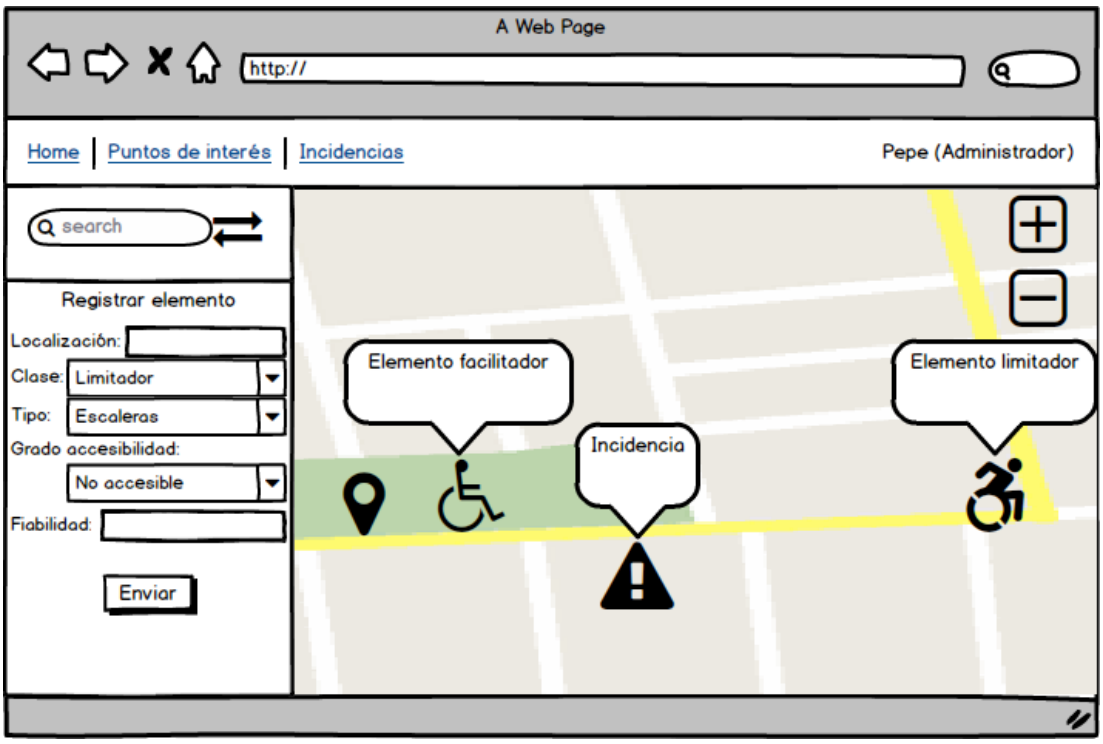


Figura A.10: Formulario elemento

Figura A.11: Incidencia detalle

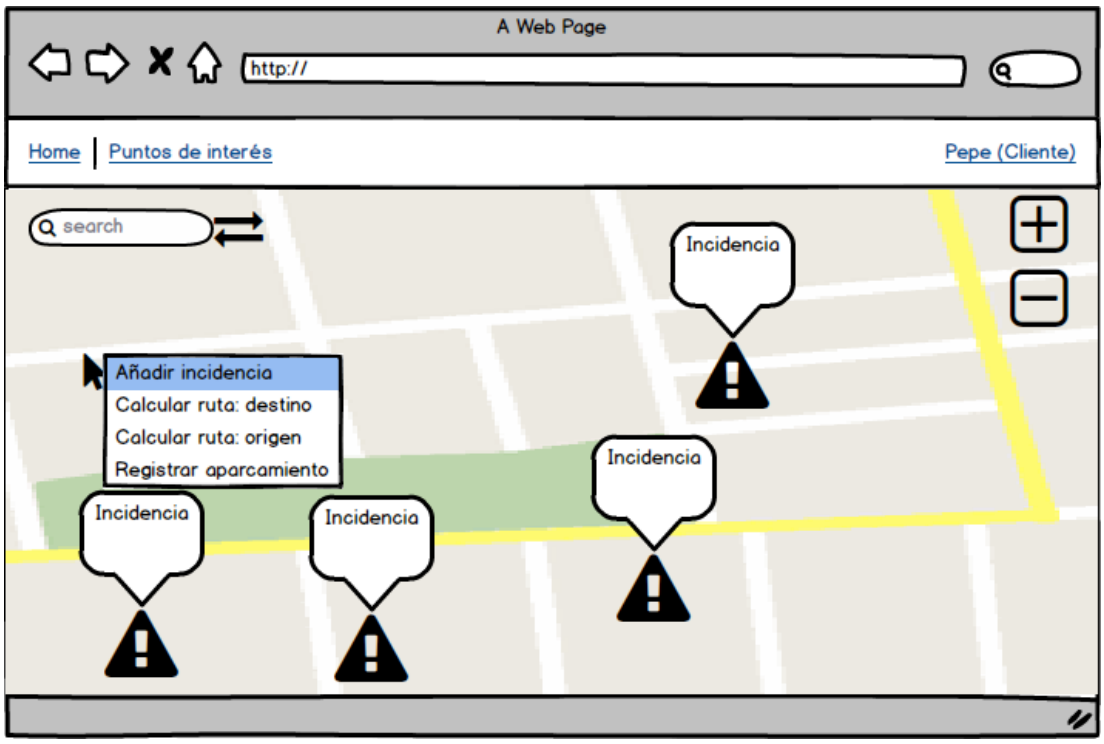


Figura A.12: Incidencias usuario

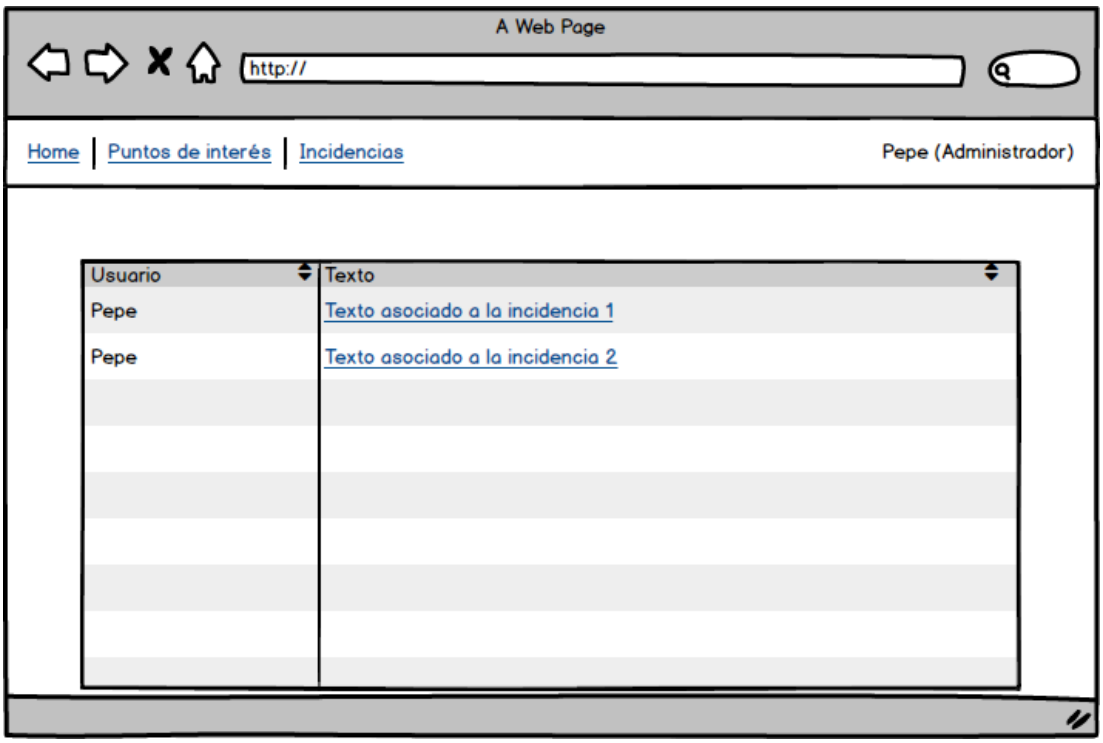


Figura A.13: Lista de incidencias

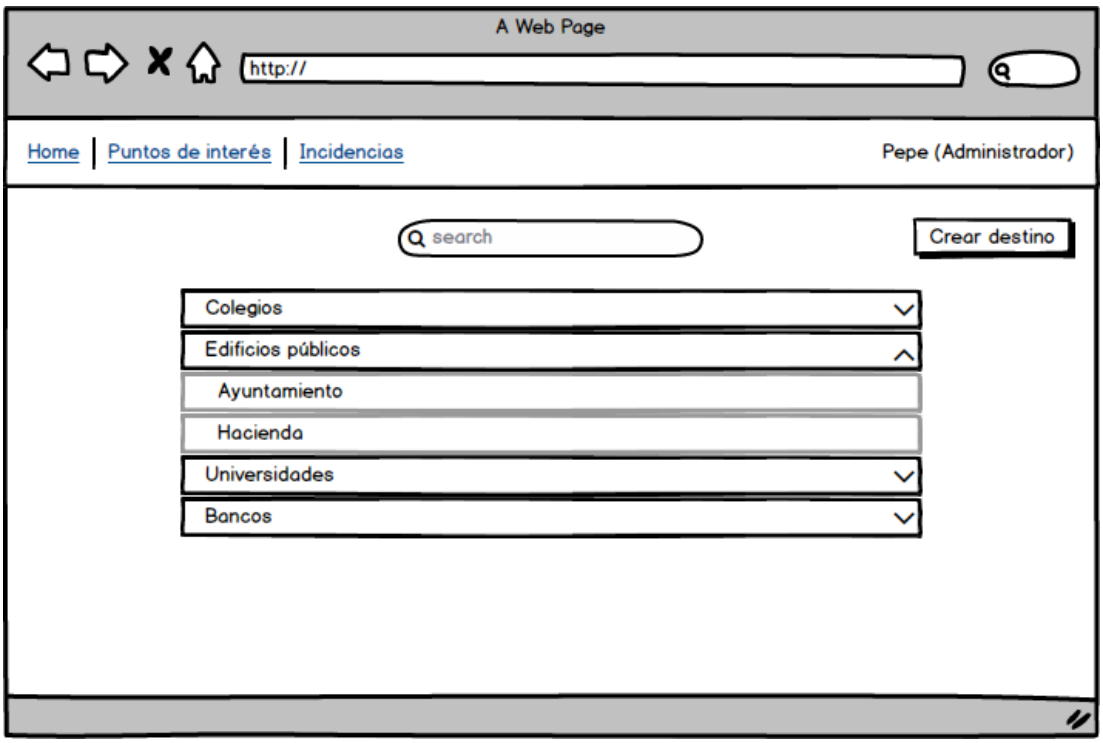


Figura A.14: Listado de destinos (administrador)

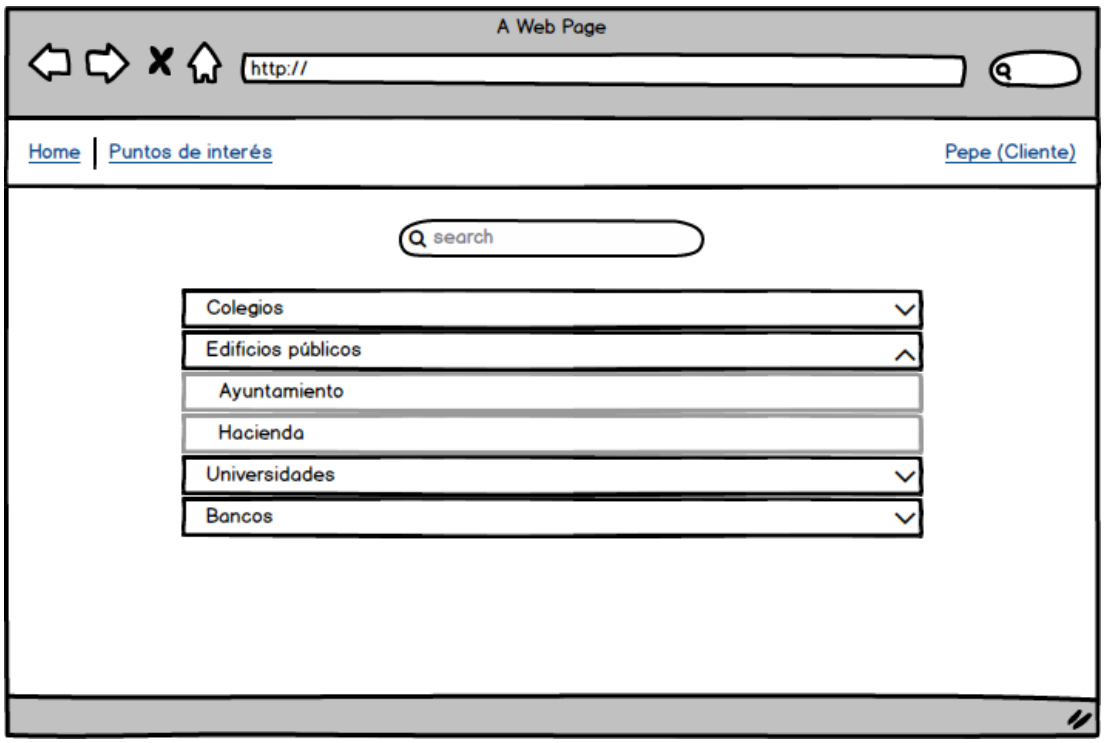


Figura A.15: Listado de destinos

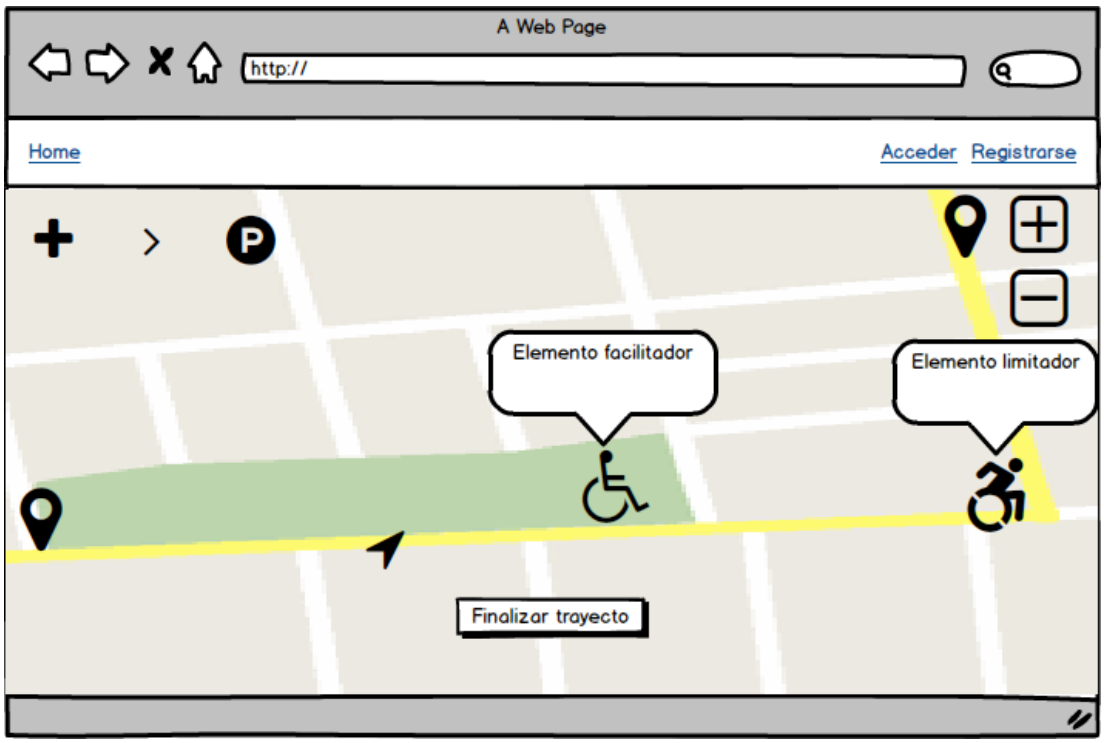


Figura A.16: Navegación

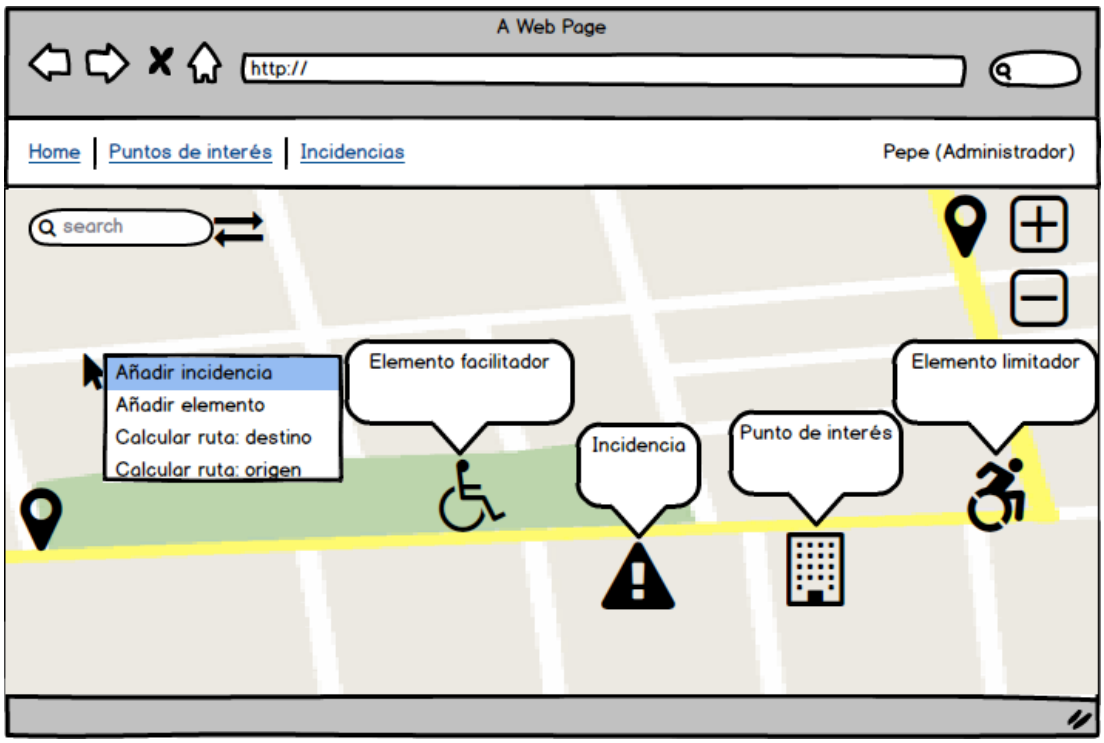


Figura A.17: Pantalla principal (administrador)

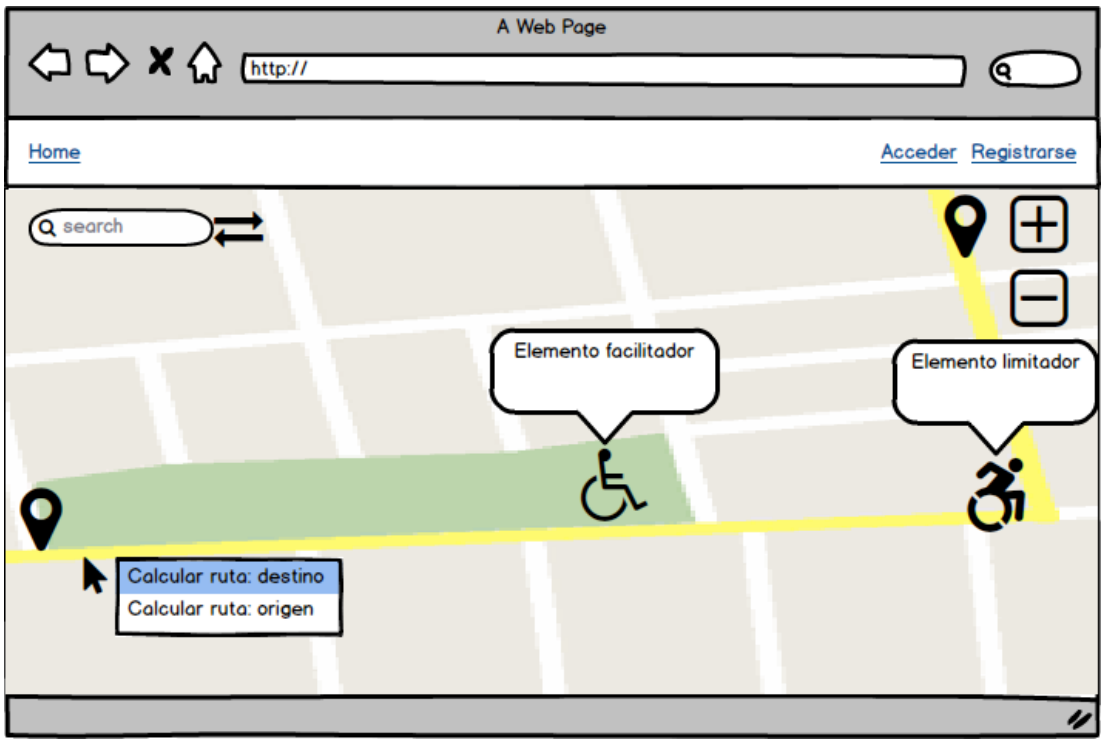


Figura A.18: Pantalla principal (anónimo)

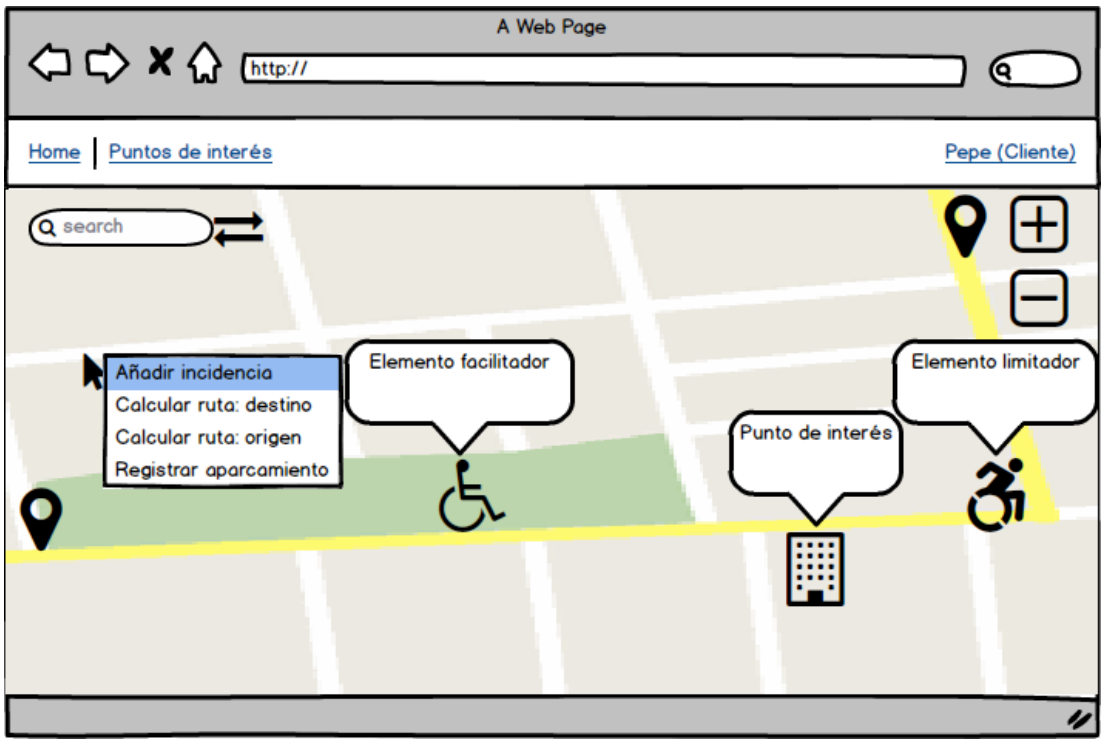


Figura A.19: Pantalla principal (cliente)

Figura A.20: Perfil de usuario

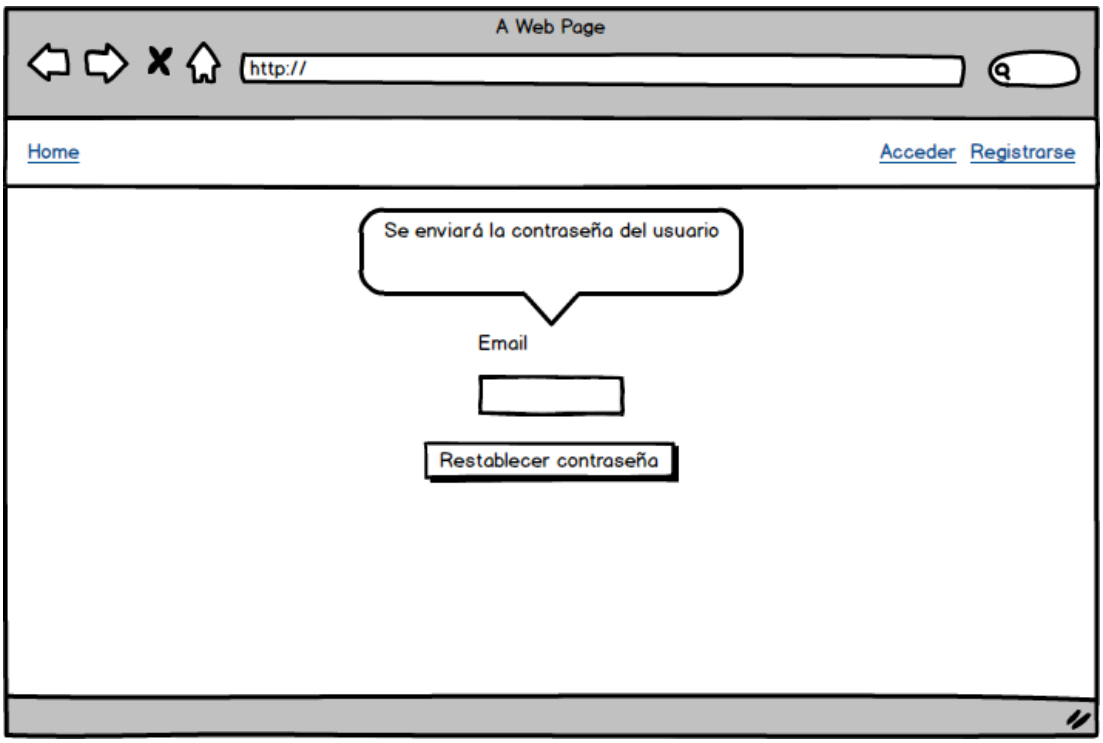


Figura A.21: Recuperar contraseña

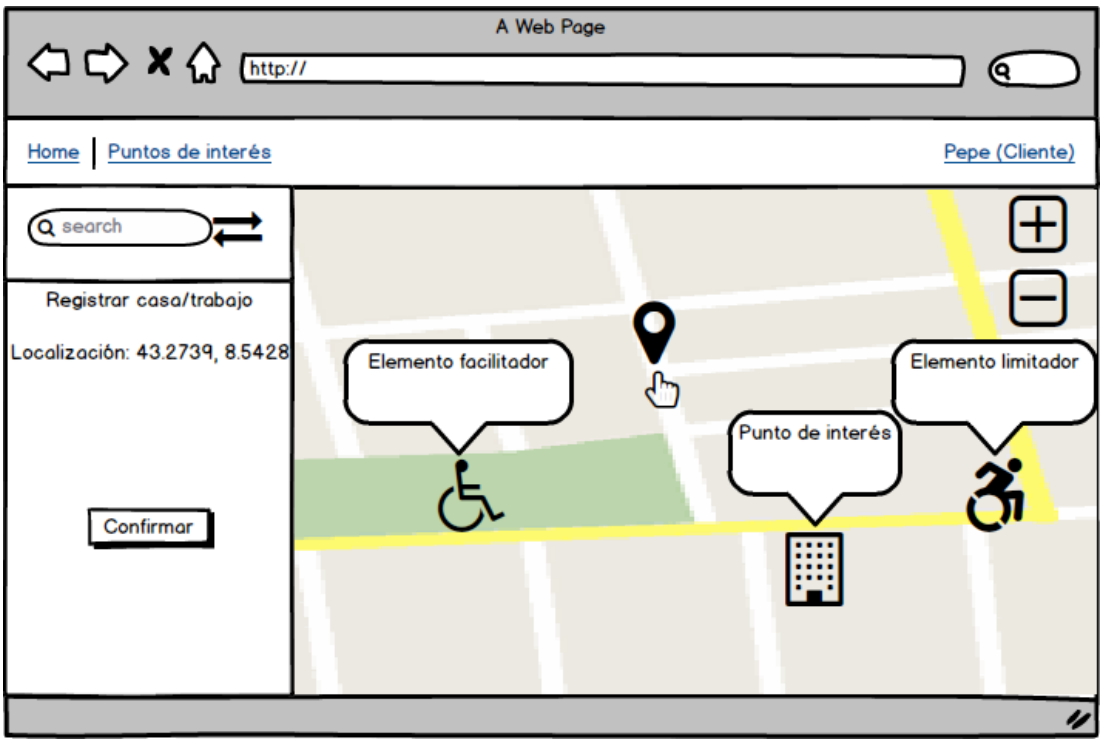


Figura A.22: Registrar casa/trabajo

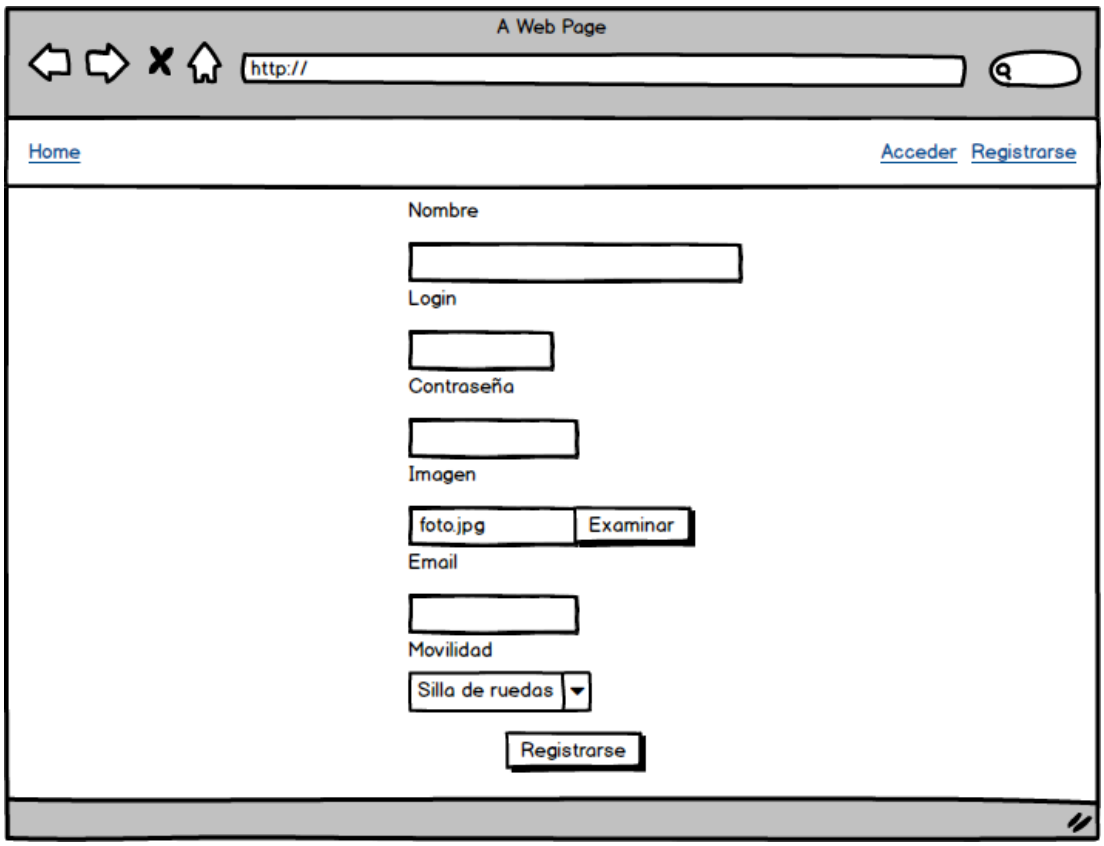


Figura A.23: Registro

Apéndice B

Manual de instalación

En este capítulo se detallan las tecnologías y pasos necesarios para el despliegue de la aplicación.

En primer lugar, es necesario tener instalado en el equipo:

- Una máquina virtual Java versión "1.8".
- Sistema de gestión de base de datos de PostgreSQL, con la extensión PostGIS instalada.
- Un servidor de aplicaciones, Apache Tomcat.
- Node.js versión 8.12.
- Apache Maven 3.3.9

Teniendo instalado todo lo mencionado, se procede a ejecutar los siguientes pasos:

- En primer lugar, se debe crear una base de datos llamada *tfg*, con un usuario llamado *asi*, con la contraseña *asi* en el puerto 5432.
- Para arrancar el servidor, se ejecuta desde el directorio *servidor-tfg* el comando *mvn spring-boot:run*. Al lanzar el servidor, se crean todas las tablas que conforman la base de datos, por lo que se deben ejecutar los ficheros *ac_2po_4pgr.sql*, *rp_2po_4pgr.sql*, *inicializarTablas.sql*, *recalcula_coste_tramo.sql* y *recalcula_coste_tramo_peatonal.sql* por este orden para cargar los datos iniciales.
- Para arrancar el cliente, se ejecuta el comando *npm install* en el directorio *cliente-tfg* y a continuación el comando *npm run debug*. Esto lanzará el cliente en la url http://localhost:1234.

Apéndice C

Glosario de acrónimos

ORM *Object-Relational mapping.*

SGBD *Sistema gestor de bases de datos.*

OSM *OpenStreetMap.*

REST *Eepresentational State Transfer.*

HTML *HyperText Markup Language.*

CSS *Cascading Style Sheets.*

JTS *Java Topology Suite.*

DAO *Data Access Object.*

DTO *Data Transfer Object.*

JSON *JavaScript Object Notation.*

HTTP *Hypertext Transfer Protocol.*

CRUD *Create Read Update Delete.*

URL *Uniform Resource Locator.*

IDE *Integrated Development Environment.*

XML *Extensible Markup Language.*

Apéndice D

Glosario de termos

Framework Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Front-controller Patrón de diseño software basado en un controlador que actúa como fachada de los controladores de un servicio web, dirigiendo todas las peticiones que llegan al servicio a los controladores específicos de este.

Backend Parte de una aplicación que se encarga del acceso a los datos y la seguridad de la información.

Feature Elemento geográfico en formato GeoJSON formado por una geometría y un apartado de propiedades.

Geometry collection Elemento geográfico en formato GeoJSON formado por una o varias geometrías y un apartado de propiedades.